SECURE DELAY OF HASH OUTPUT

A secure delay according to various aspects of the present inventions can be applied in other areas than just passphrase security. For example, a hash value can be run through a secure delay to produce a smaller hash value that would be computationally infeasible to derive based on a birthday attack. In one conceived embodiment, a 160-bit hash value is repeatedly run through a secure delay for a predetermined number of iterations that, given a security selection, corresponds to an acceptable unit delay. (An example of an acceptable unit delay is 1 second.) At the end of each unit delay, a sub-hash is computed from the current output of the secure delay and displayed.

A person wishing to compare hash values can begin comparing a first group of digits corresponding to the first sub-hash after the first unit delay, and while the second unit delay is underway. When the person looks for the second group of digits to compare, the second unit delay (when optimally chosen) is already completed and the third unit delay is underway.

Thus, a securely delayed hash system according to various aspects of the inventions can provide a smaller hash value with the same security as the larger hash value from which it is derived. The loss in entropy in the smaller value is offset by the computational difficulty (from the secure delay) of obtaining the smaller value. An attacker wishing to find a larger hash value that produces the smaller hash value will need to run the secure delay, on average, N2/2 times with the secure delay computation for each iteration.

If T = delay time (on an equivalent processor as that of the legitimate user), then T2 = T*N2/2. If T = 1 second, and the required T2 = 1,000,000 CPU years, then the required N2 ~= 2^21, a much smaller value than, say, 2^160.

Since a hash value is not particularly sensitive, it can be sent freely over in secure networks. It is conceivable that an Internet site can be established for quickly computing smaller "sub-hashes" based on transmitted hash values through an open-source, standardized secure delay algorithm. However, it is more likely that the market will demand simplicity and standardization, and an average delay within the reach of the average desktop PC. (The remote-computation model may be useful for portable computers, though.)

# # #

Choices : $7 \times 5 = 35 = M$

Digits : $N = 8$

No duplicate digits

$\therefore$

$$X = M \cdot (M-1) \cdot (M-2) \cdot (M-3) \ldots (M-N$$

$$X = \frac{M!}{(M-N-2)!} = \frac{35!}{25!}$$

$$X = 6.67 \times 10^{14}$$

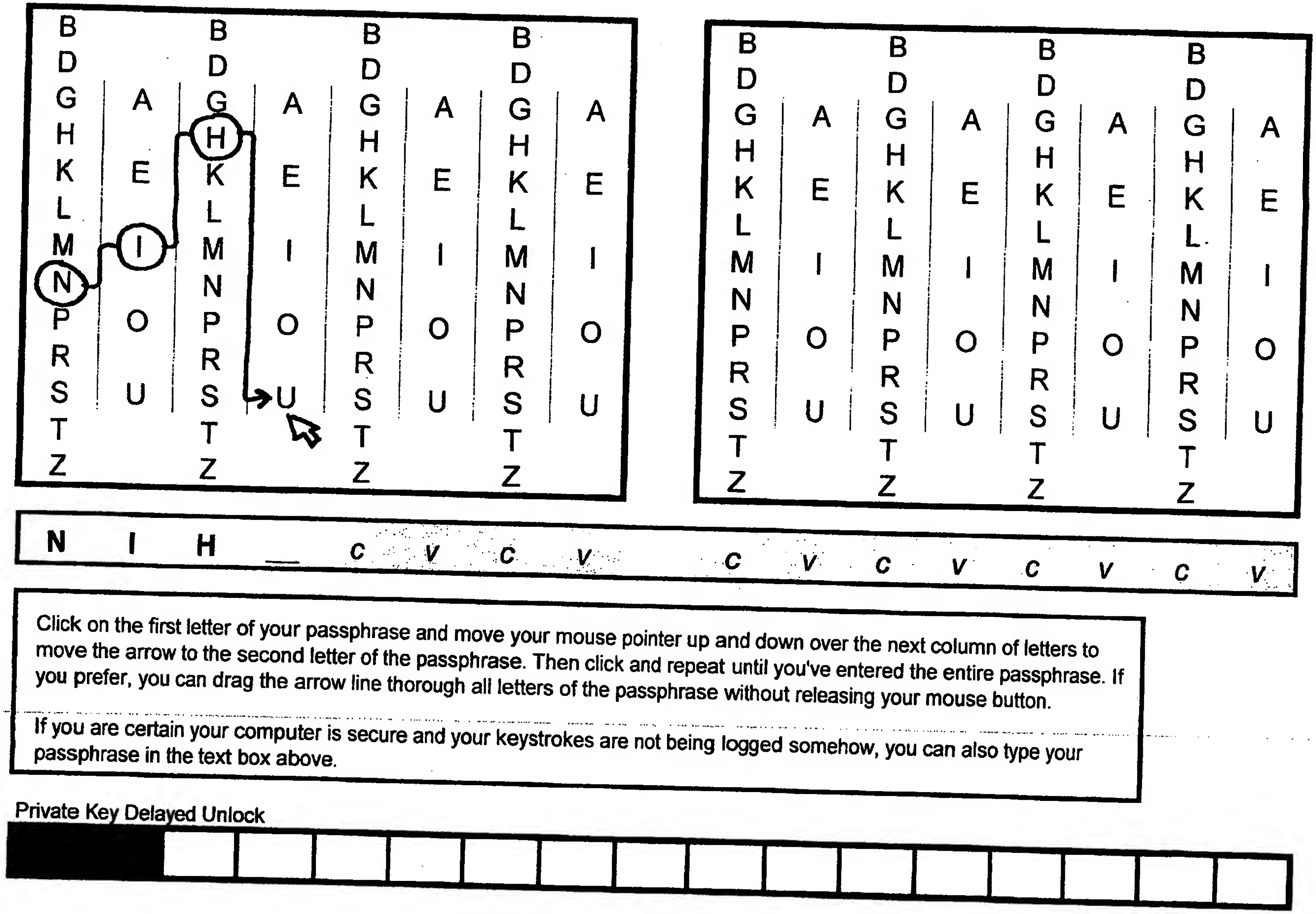
$\log_2(X) = 49.2$ bits of entropy

With 0.5 sec. delay between digits,
$d = 4$ sec

$$\frac{4X}{(\underset{\uparrow \atop \text{sec/hr}}{3600} \times \underset{\uparrow \atop \text{hr/d}}{24} \times \underset{\uparrow \atop \text{d/yr}}{365})} = 84{,}496{,}818 \text{ yrs } (!)$$

on comparable delay processor
w/ 2x every 18 mos, in
20 yrs : 8,187 yrs (!)

AB-1


B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U

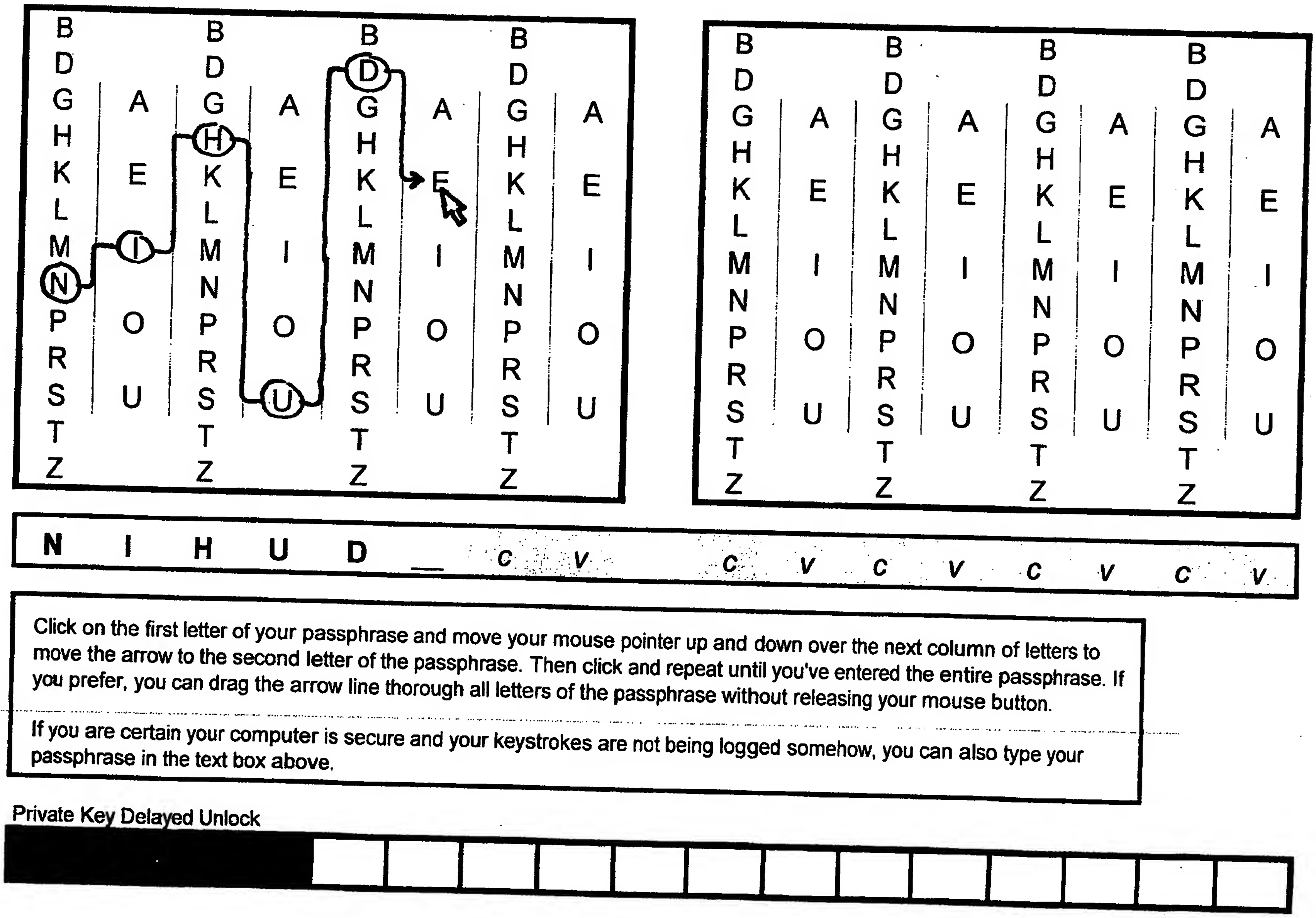
N I H __ c v c v c v c v c v c v

Click on the first letter of your passphrase and move your mouse pointer up and down over the next column of letters to move the arrow to the second letter of the passphrase. Then click and repeat until you've entered the entire passphrase. If you prefer, you can drag the arrow line thorough all letters of the passphrase without releasing your mouse button.

If you are certain your computer is secure and your keystrokes are not being logged somehow, you can also type your passphrase in the text box above.

Private Key Delayed Unlock

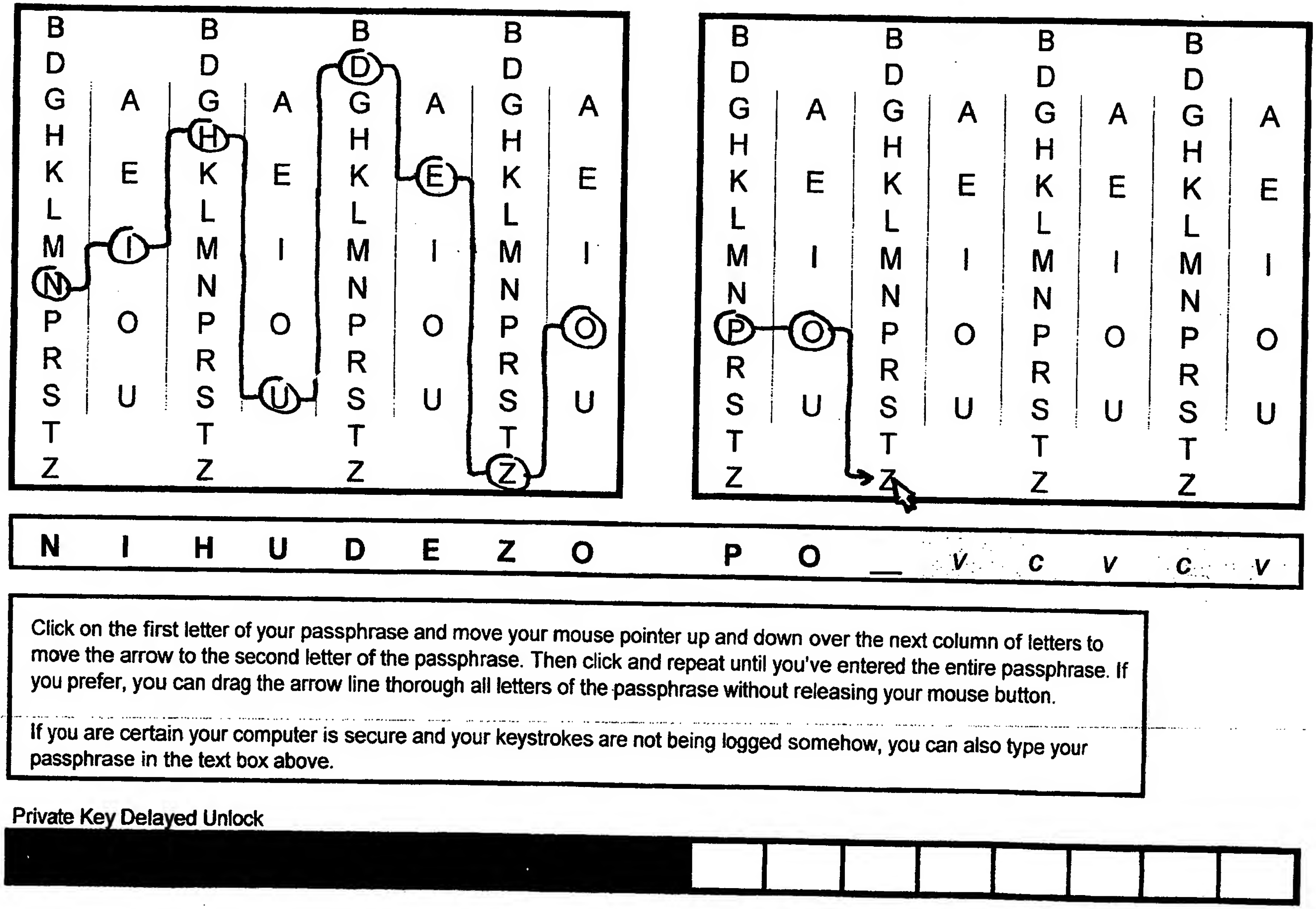
Z-2


B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
N I H U D _ c v c v c v c v c v

Click on the first letter of your passphrase and move your mouse pointer up and down over the next column of letters to move the arrow to the second letter of the passphrase. Then click and repeat until you've entered the entire passphrase. If you prefer, you can drag the arrow line thorough all letters of the passphrase without releasing your mouse button.

If you are certain your computer is secure and your keystrokes are not being logged somehow, you can also type your passphrase in the text box above.

Private Key Delayed Unlock


B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U

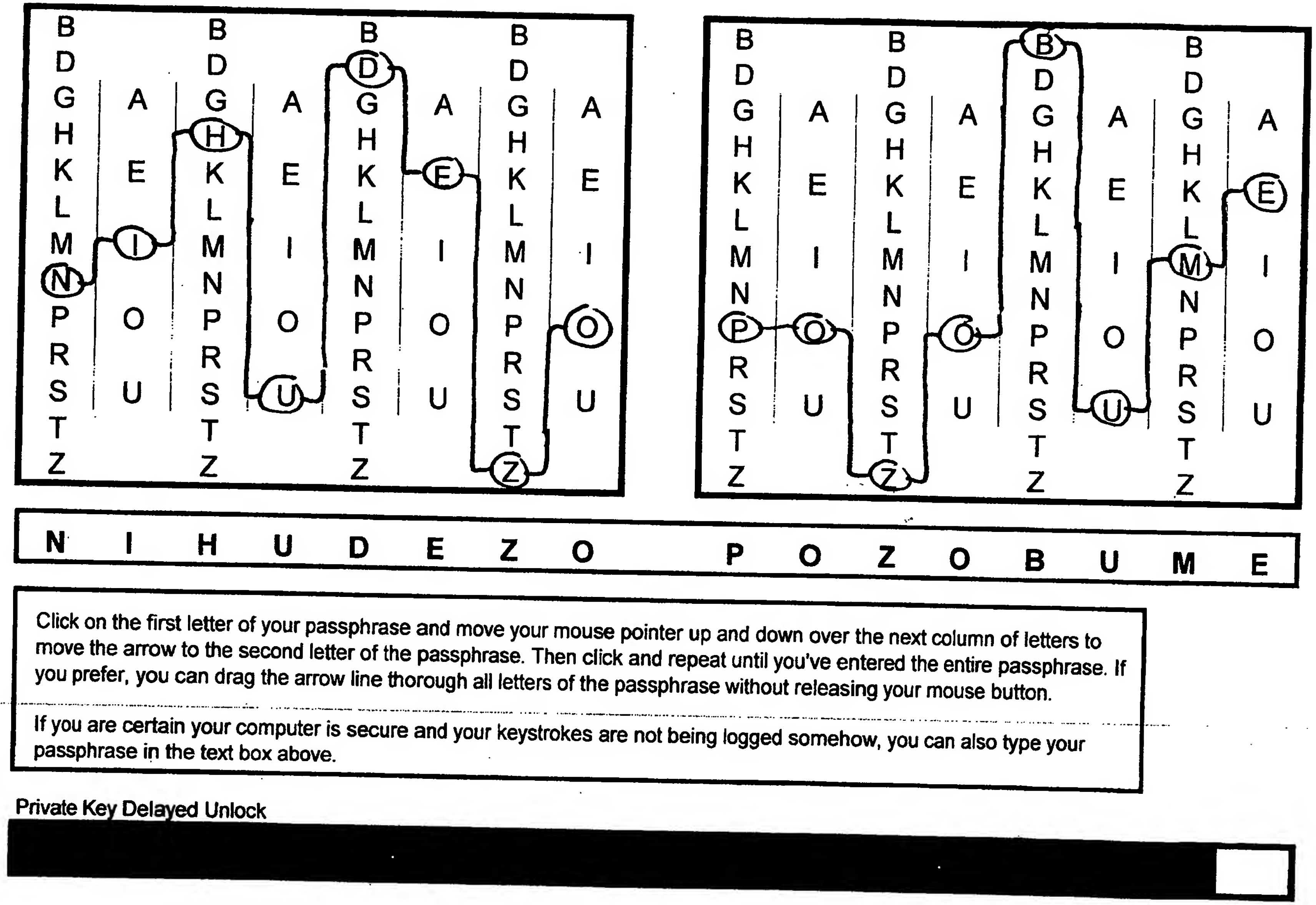
N I H U D E Z O P O __ V C V C V

Click on the first letter of your passphrase and move your mouse pointer up and down over the next column of letters to move the arrow to the second letter of the passphrase. Then click and repeat until you've entered the entire passphrase. If you prefer, you can drag the arrow line thorough all letters of the passphrase without releasing your mouse button.

If you are certain your computer is secure and your keystrokes are not being logged somehow, you can also type your passphrase in the text box above.

Private Key Delayed Unlock

Z-4

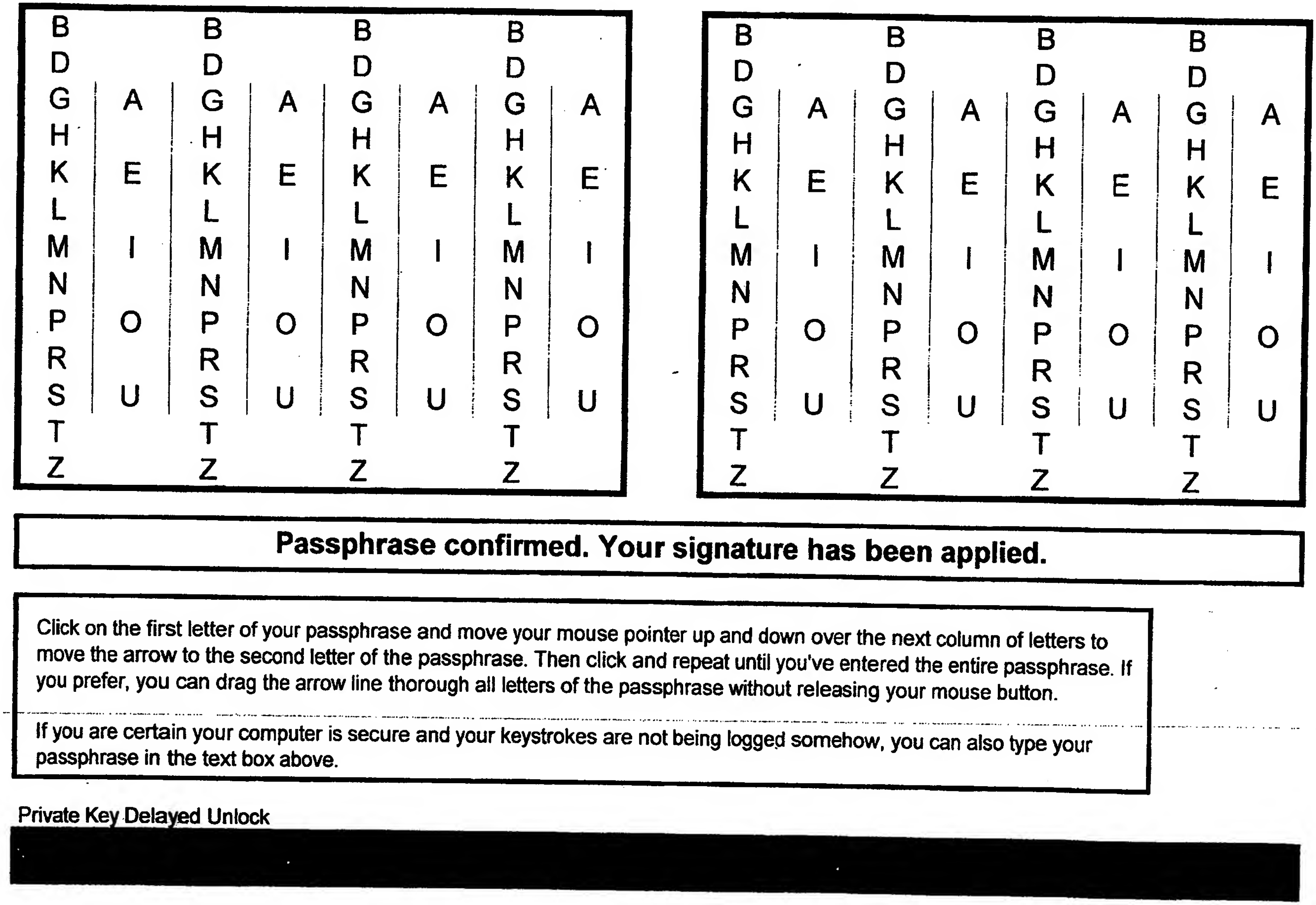

B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U
B D G H K L M N P R S T Z
A E I O U

N I H U D E Z O P O Z O B U M E

Click on the first letter of your passphrase and move your mouse pointer up and down over the next column of letters to move the arrow to the second letter of the passphrase. Then click and repeat until you've entered the entire passphrase. If you prefer, you can drag the arrow line thorough all letters of the passphrase without releasing your mouse button.

If you are certain your computer is secure and your keystrokes are not being logged somehow, you can also type your passphrase in the text box above.

Private Key Delayed Unlock



B		B		B		B	
D		D		D		D	
G	A	G	A	G	A	G	A
H		H		H		H	
K	E	K	E	K	E	K	E
L		L		L		L	
M	I	M	I	M	I	M	I
N		N		N		N	
P	O	P	O	P	O	P	O
R		R		R		R	
S	U	S	U	S	U	S	U
T		T		T		T	
Z		Z		Z		Z	

B		B		B		B	
D		D		D		D	
G	A	G	A	G	A	G	A
H		H		H		H	
K	E	K	E	K	E	K	E
L		L		L		L	
M	I	M	I	M	I	M	I
N		N		N		N	
P	O	P	O	P	O	P	O
R		R		R		R	
S	U	S	U	S	U	S	U
T		T		T		T	
Z		Z		Z		Z	

Passphrase confirmed. Your signature has been applied.

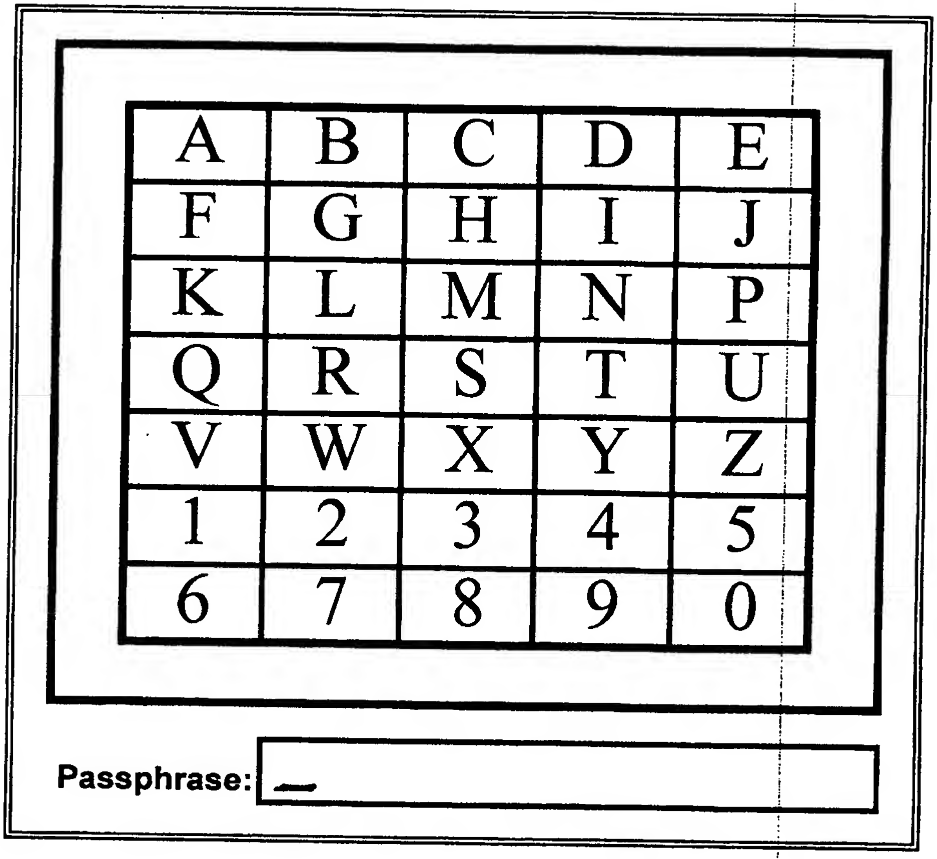
Click on the first letter of your passphrase and move your mouse pointer up and down over the next column of letters to move the arrow to the second letter of the passphrase. Then click and repeat until you've entered the entire passphrase. If you prefer, you can drag the arrow line thorough all letters of the passphrase without releasing your mouse button.

If you are certain your computer is secure and your keystrokes are not being logged somehow, you can also type your passphrase in the text box above.

Private Key Delayed Unlock

[redacted]

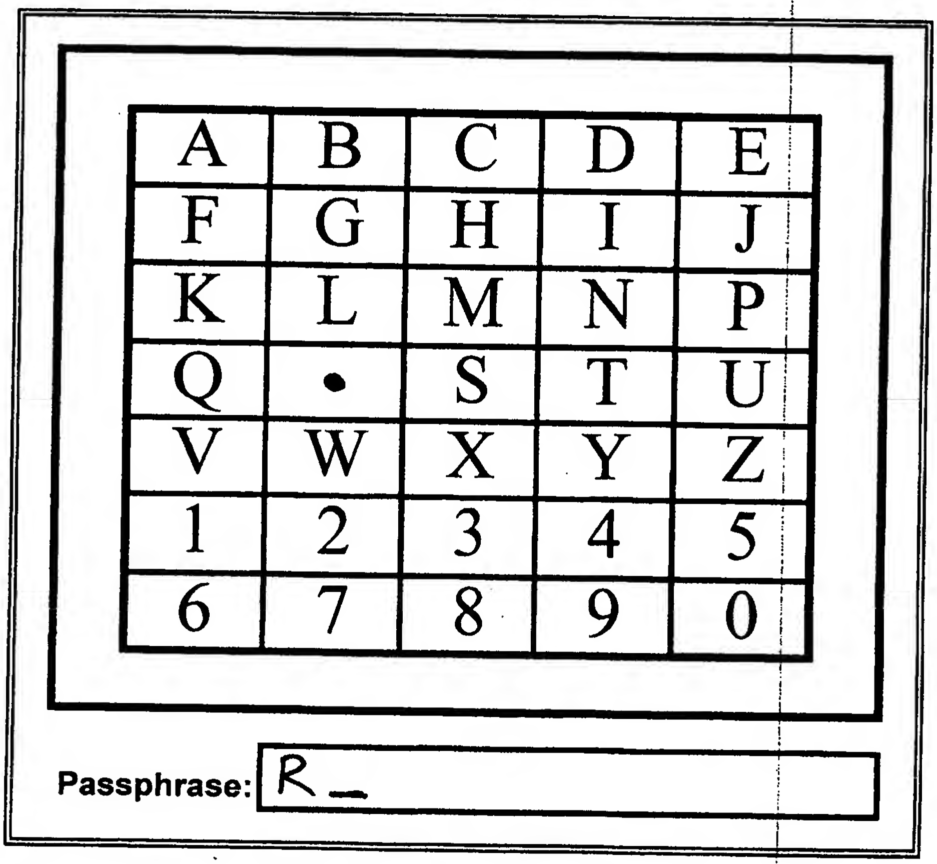
Z-6



A	B	C	D	E
F	G	H	I	J
K	L	M	N	P
Q	R	S	T	U
V	W	X	Y	Z
1	2	3	4	5
6	7	8	9	0

Passphrase: _

In a variation, entry system is built into keypad of USB-connected hardware delay processor

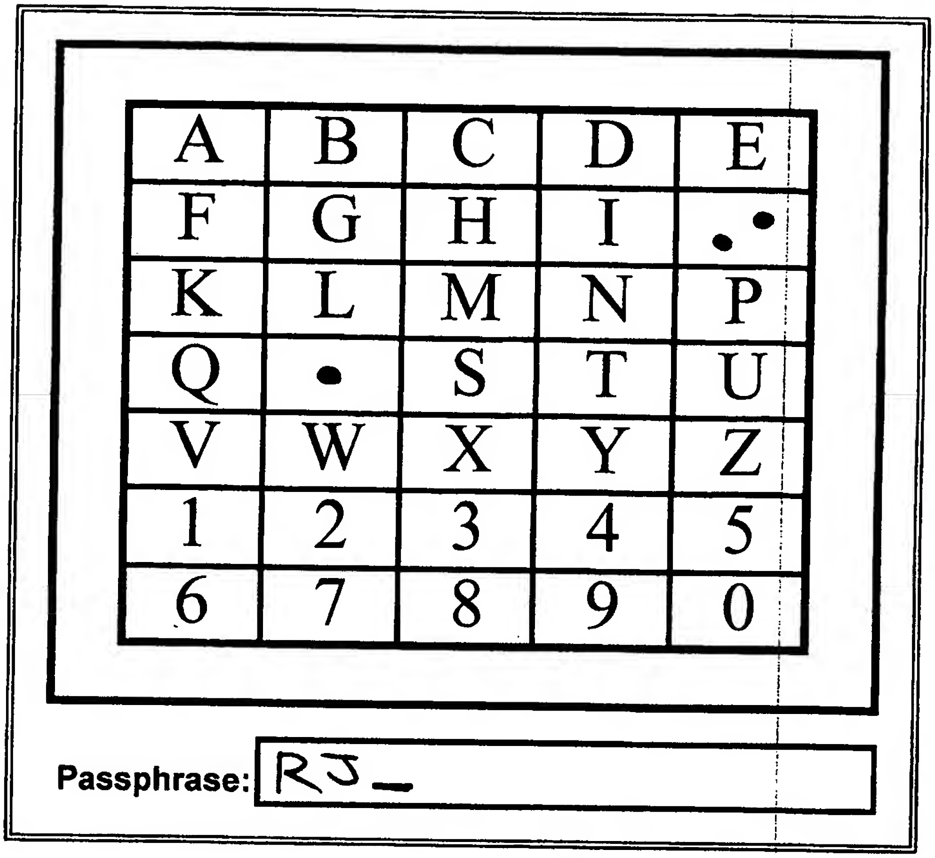


A	B	C	D	E
F	G	H	I	J
K	L	M	N	P
Q	●	S	T	U
V	W	X	Y	Z
1	2	3	4	5
6	7	8	9	0

Passphrase: R_

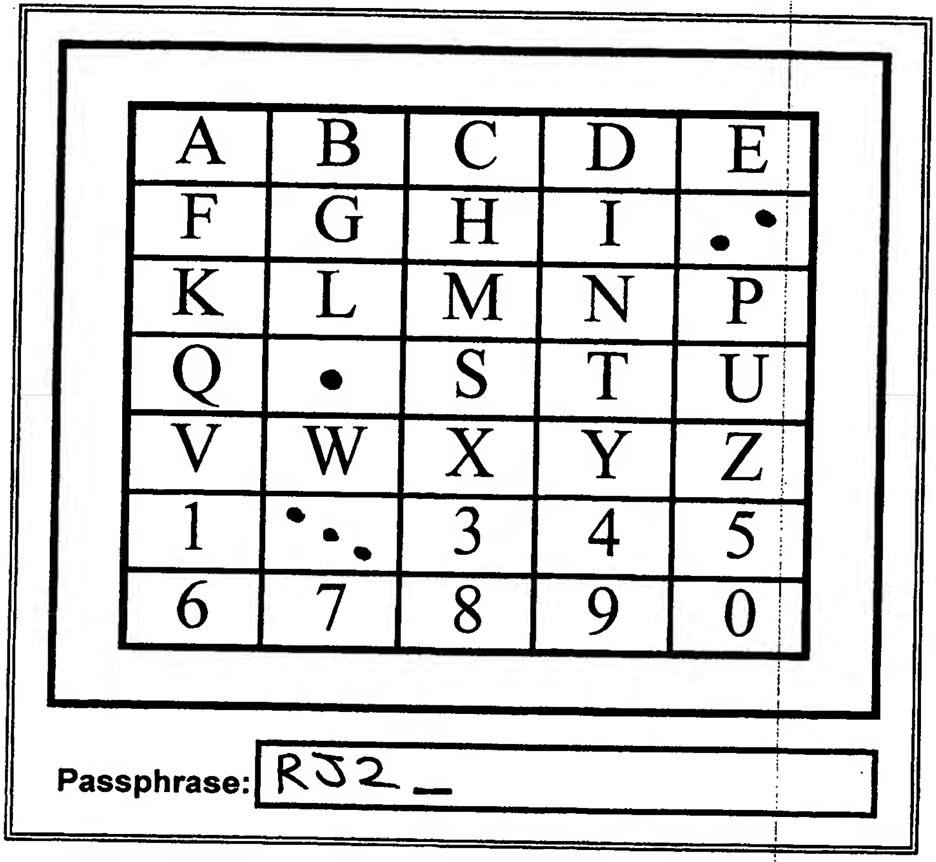
Already-selected digits can be illuminated

AB-2



A	B	C	D	E
F	G	H	I	• •
K	L	M	N	P
Q	•	S	T	U
V	W	X	Y	Z
1	2	3	4	5
6	7	8	9	0

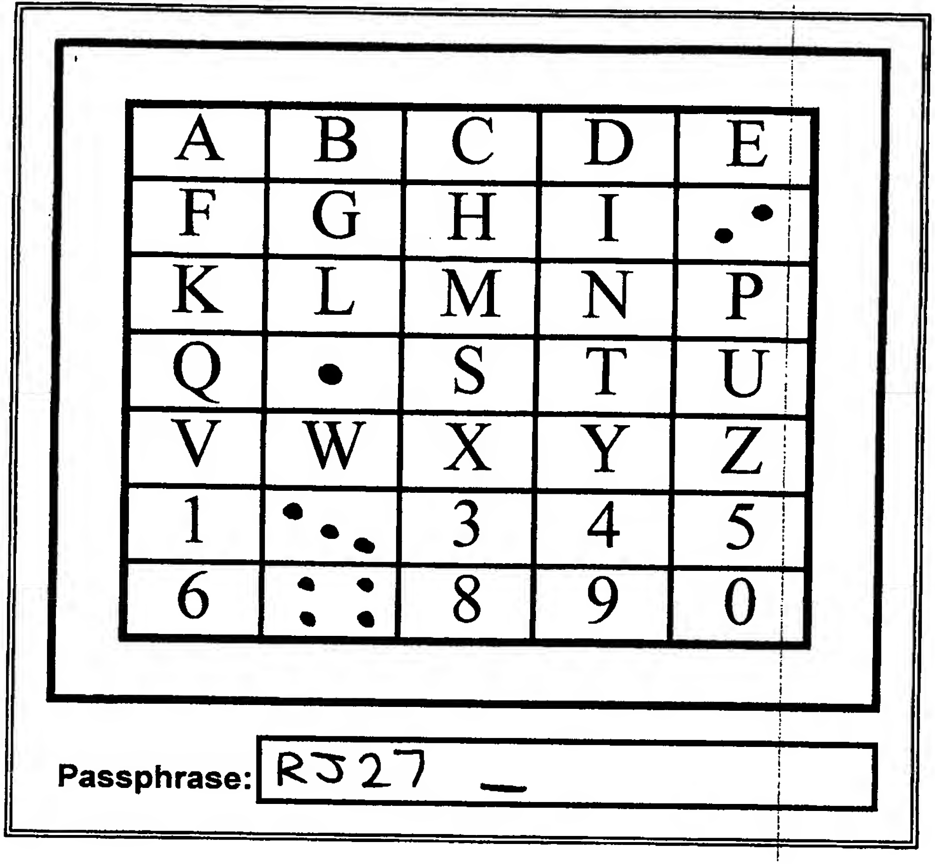
Passphrase: RJ_



A	B	C	D	E
F	G	H	I	• •
K	L	M	N	P
Q	•	S	T	U
V	W	X	Y	Z
1	• • •	3	4	5
6	7	8	9	0

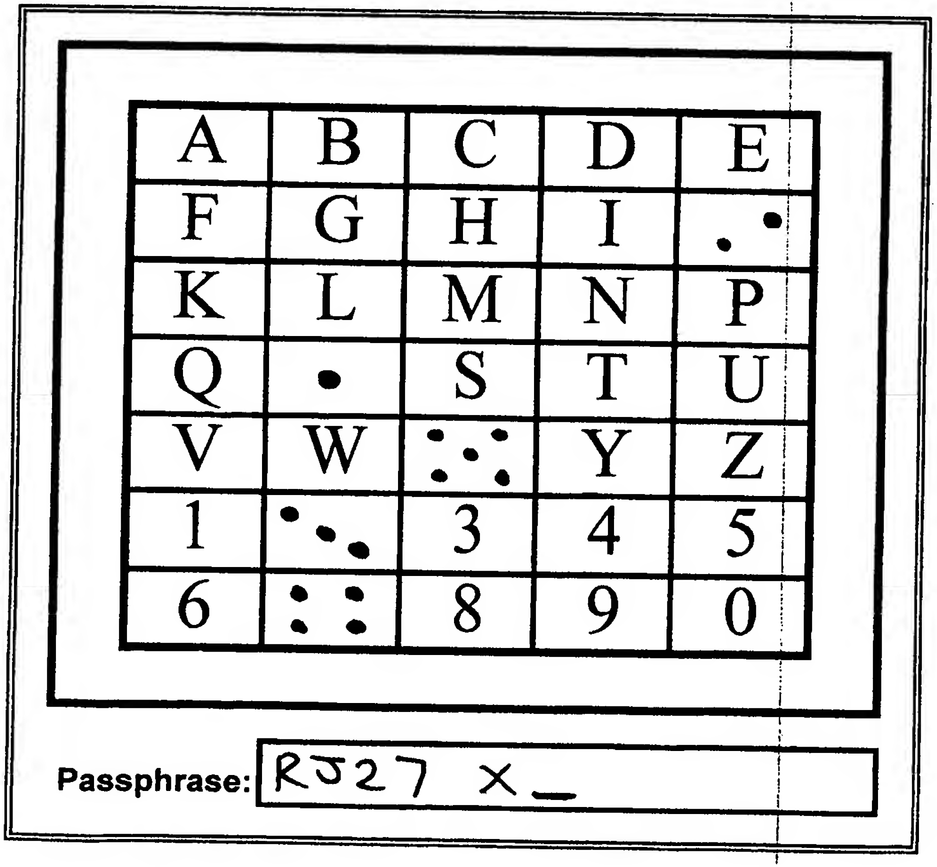
Passphrase: RJ2_

AB-3



A	B	C	D	E
F	G	H	I	●●
K	L	M	N	P
Q	●	S	T	U
V	W	X	Y	Z
1	●●●	3	4	5
6	●●●●	8	9	0

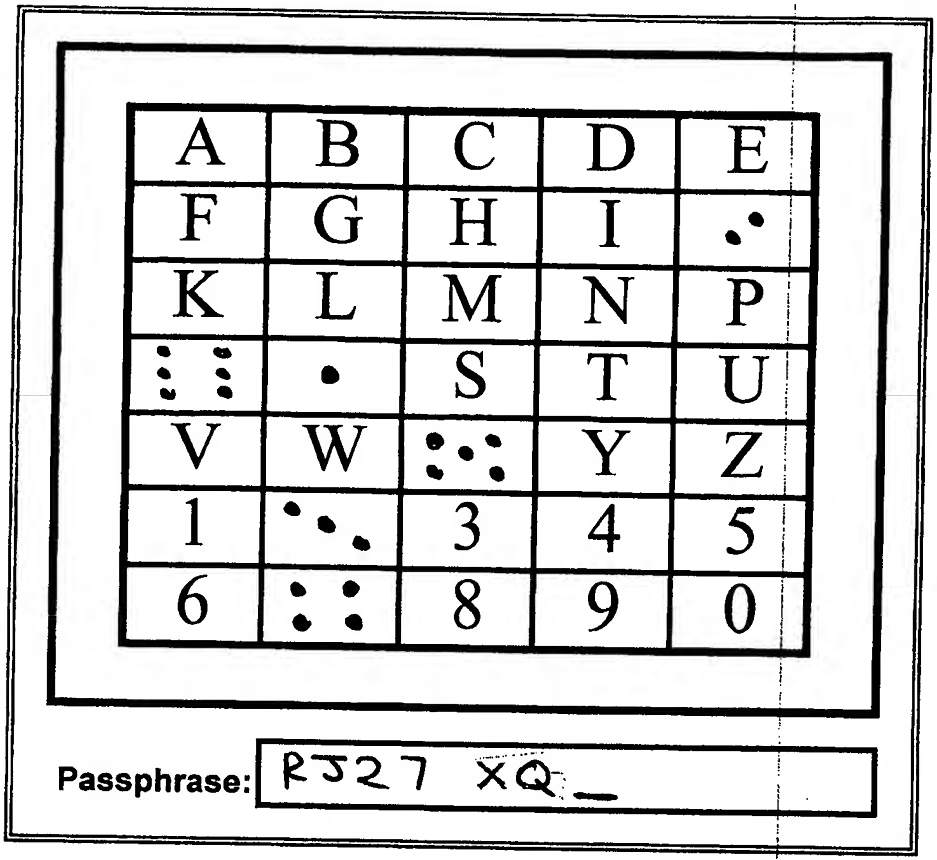
Passphrase: RJ27 _



A	B	C	D	E
F	G	H	I	●●
K	L	M	N	P
Q	●	S	T	U
V	W	●●●●●	Y	Z
1	●●●	3	4	5
6	●●●●	8	9	0

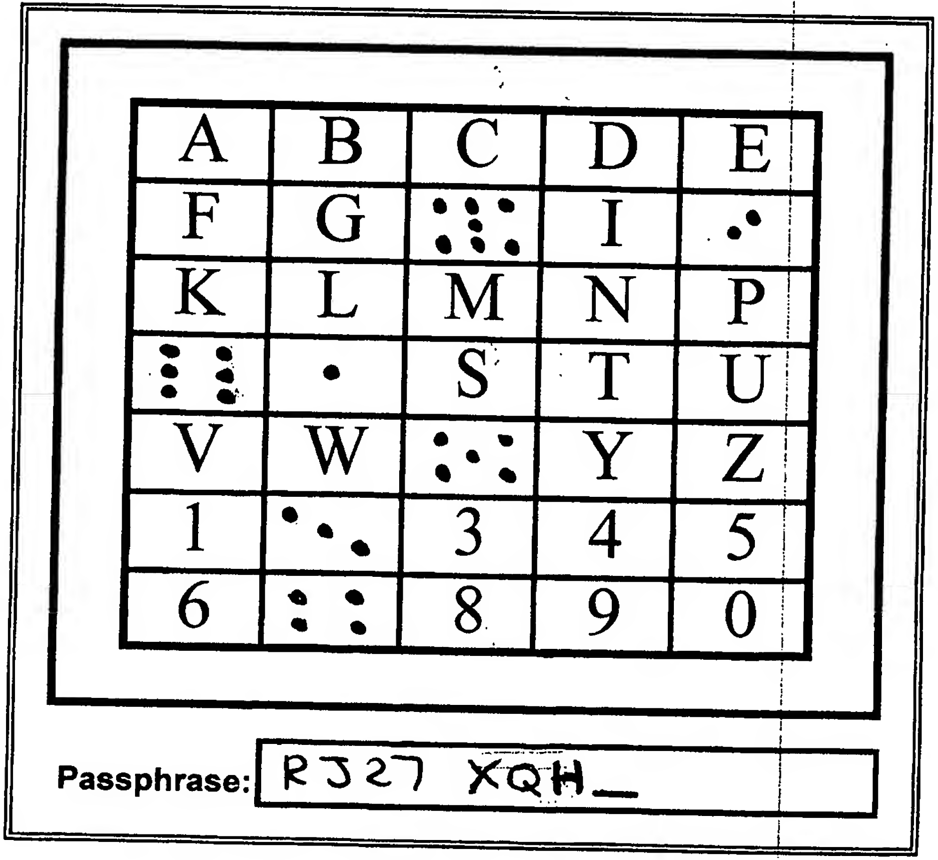
Passphrase: RJ27 X_

AB-4



A	B	C	D	E
F	G	H	I	⚁
K	L	M	N	P
⚅	⚀	S	T	U
V	W	⚄	Y	Z
1	⚂	3	4	5
6	⚃	8	9	0

Passphrase: RJ27 XQ_



A	B	C	D	E
F	G	⚅	I	⚁
K	L	M	N	P
⚅	⚀	S	T	U
V	W	⚄	Y	Z
1	⚂	3	4	5
6	⚃	8	9	0

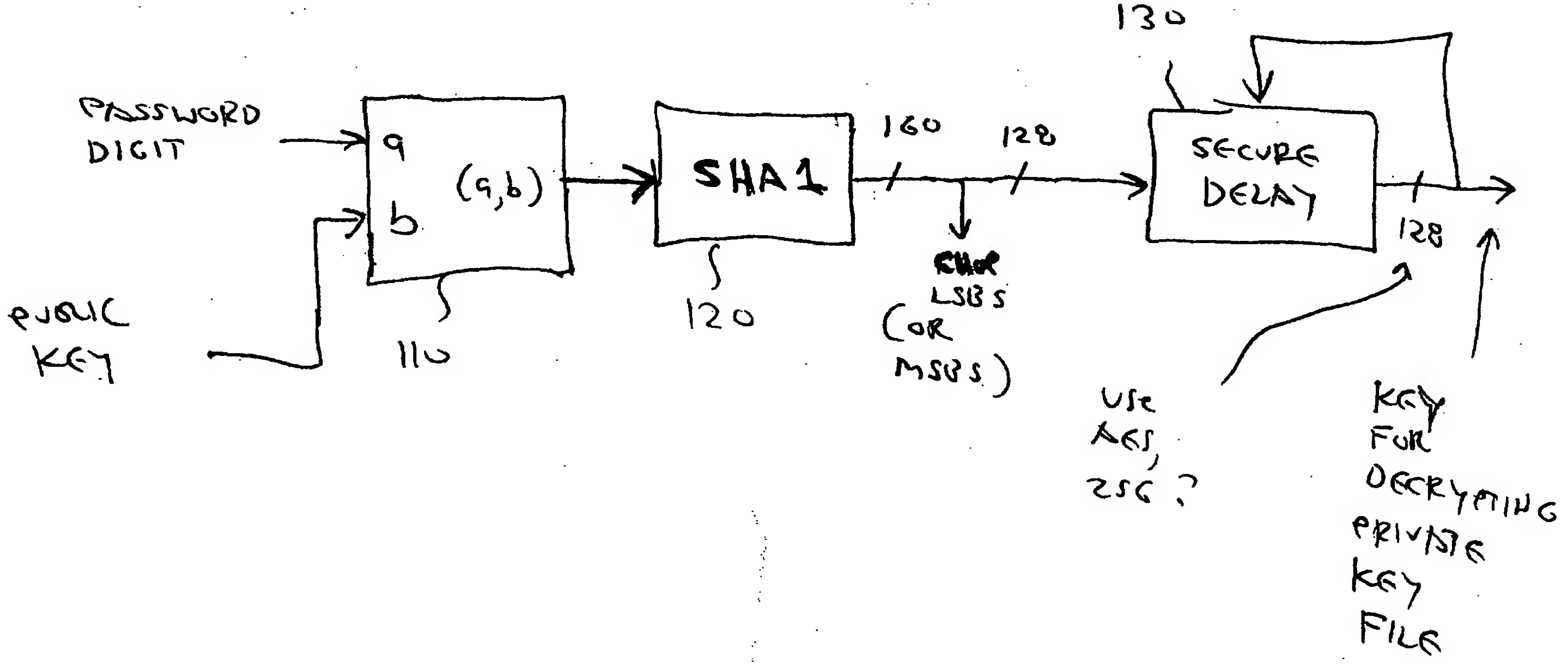
Passphrase: RJ27 XQH_

AB-5

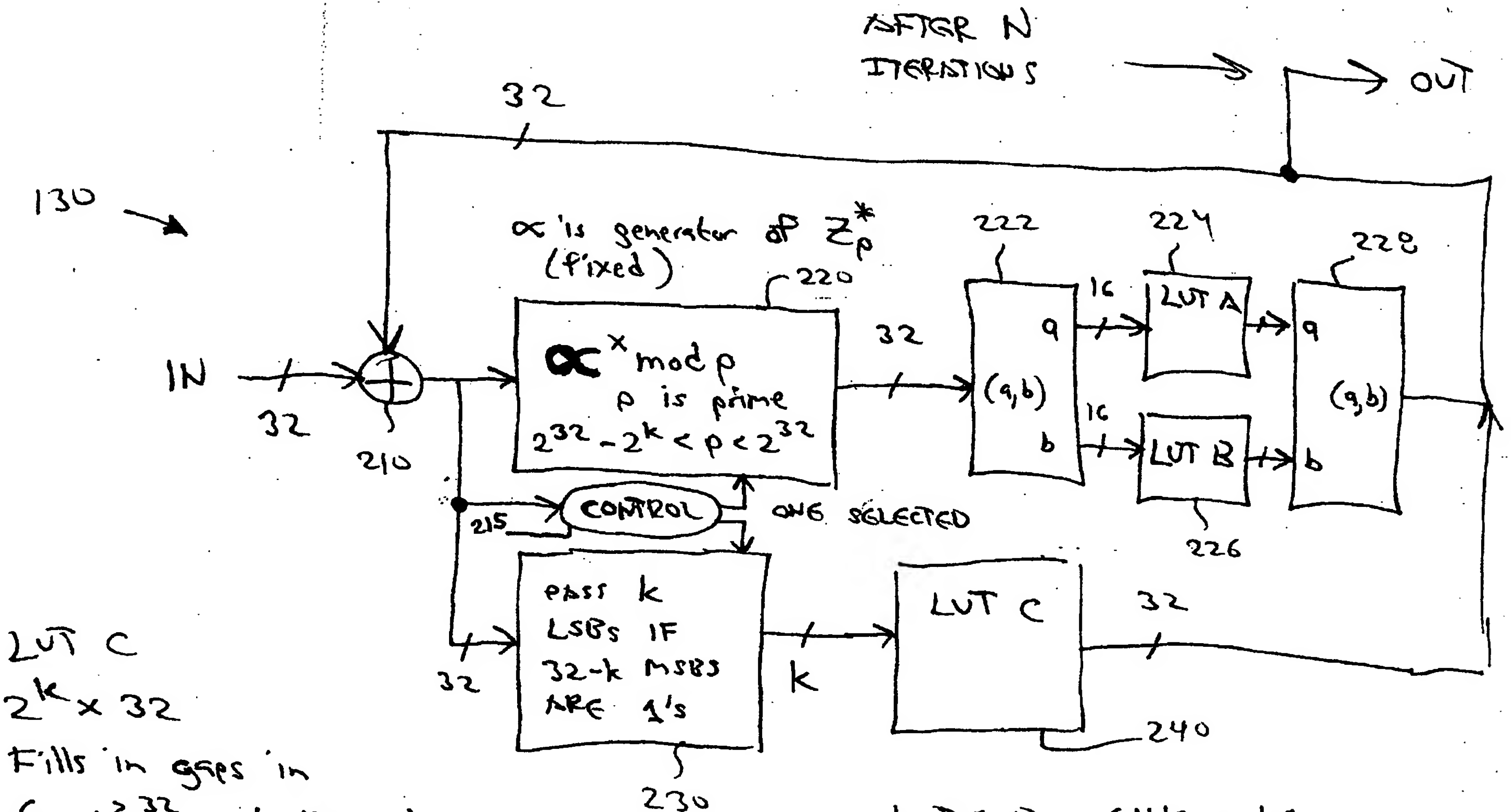
A	B	C	D	E
F	G	●●●●●●●	I	●●
K	L	M	N	P
●●●●●●	●	S	T	U
V	W	●●●●●	Y	Z
1	●●●	●●●●●●●●	4	5
6	●●●●	8	9	0

Passphrase: RJ27 XQH3

AB-6


PASSWORD DIGIT
PUBLIC KEY
a
b
(a,b)
110
SHA1
120
160
128
LSBS (OR MSBS)
130
SECURE DELAY
128
USE AES, 256 ?
KEY FOR DECRYPTING PRIVATE KEY FILE

AC-1



LUT C

$2^k \times 32$

Fills in gaps in $\{0,1\}^{32}$ not reached by LUTS (A,B) because $p < 2^{32}$, $\alpha^x \bmod p$ not selected for $x \geq 2^{32} - 2^k$.

ASSUMES

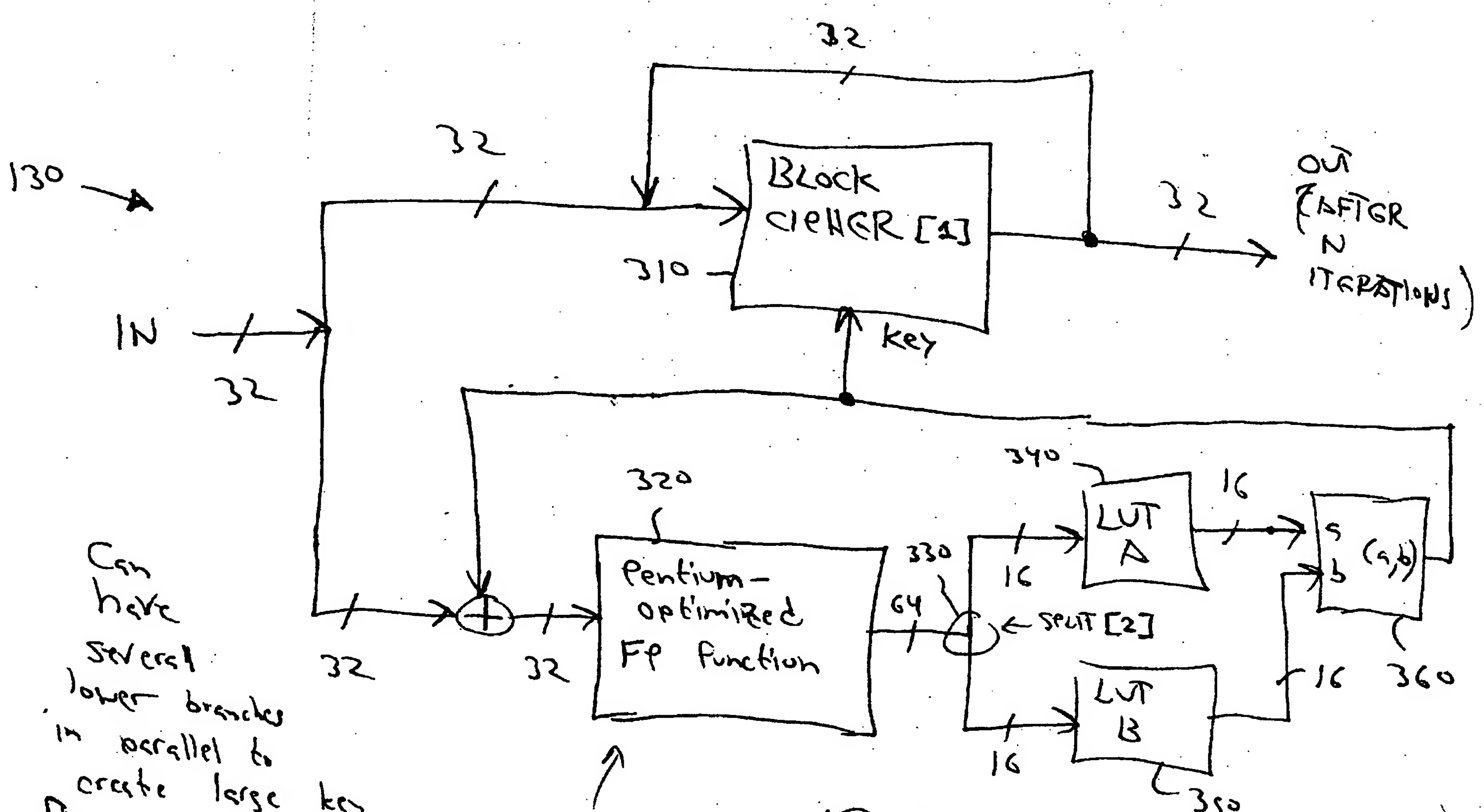
THERE IS A PROVABLE PRIME $< 2^{32}$ and within 2^k of 2^{32}

k small (4-5)

LUT A,B 64K × 16

Can be same random mappings of $\{0,1\}^{16}$ or different.

ONE EXEMPLARY EMBODIMENT OF 32-BIT SECURE DELAY



Can have several lower branches in parallel to create large key for single block cipher.

SIN(x), tan(x), etc. ← CAN CYCLE THRU DIFFERENT FUNCTIONS

[1] SHOULD BE FAST, SIMPLE SO FP FUNCTION DOMINATES PROCESSING (MAYBE JUST XOR?)

[2] SPLIT: SEND: BITS 48-63 TO LUT A
BITS 32-47 TO LUT B

ANOTHER EMBODIMENT OF 32-BIT SECURE DELAY

VIRTUAL SIGNATURE PRINTER

According to various aspects of the present inventions, an electronic record (e.g., MS Word document, AUTOCAD drawing, etc.) can be signed by "printing" that record to a special "virtual signature printer." An example of a virtual printer is the "PDF Writer" printer driver that is installed with the ADOBE ACROBAT software. The inventive "virtual signature printer" provides the user with an intuitive, simple way of authenticating an electronic record.

The "virtual signature printer" system produces an output file (or multiple files, see below) that can be sent to a recipient for viewing, printing, and validation. The recipient can view or print the file (preferably, the file is "backward compatible" compatible with widely available viewing software) and, with special software, can validate the signature on the file.

A particularly advantageous way of signing an electronic record that has been "printed" this way is with embedded signatures. However, a "virtual signature printer" system can employ any suitable technique for signing an electronic record. The following are some examples of output of such a system:

- A PS or TIFF file (or, with suitable licensing if necessary, a PDF file) representing the document, accompanied by a detached PGP-compatible signature file.
- A ZIP file containing a PS, TIFF, or PDF file representing the document including a ZIP comment containing a Base-64 PGP-compatible signature.
- A PGP-signed file containing the document in a PS, TIFF, or PDF file.

When the signer wishes to electronically sign a document he or she can print the document to the virtual signature printer driver, using the print functionality of the software used to create the document. The printer driver creates a window in which the software requests the signer's authenticating information. The user can enter his or her passphrase, apply his or her fingerprint to a fingerprint scanner, insert a smart card, etc. The software then computes the digital signature for the document, based on the authenticating information or a private key unlocked by the authenticating information, and embeds the digital signature with an output file or includes the digital signature in a separate file.

# # #

EDWIN A. SUOMINEN

U.S. Patents
5,926,513; 5,937,341*;
6,052,748*; 6,069,913
Additional patents pending*

*Available for licensing

14614 North Kierland Boulevard,
Suite 300
Scottsdale, Arizona 85254

Telephone: (480) 948-3295
Facsimile: (480) 948-3387

Registered Patent Agent
Independent Inventor of Electrical
Engineering Technology

www.eepatents.com
ed@eepatents.com

EXAMPLE

VIA EMAIL
Thomas E. Workman, Jr., Esq.
Law Offices of Thomas Workman
41 Harrison Street
Taunton, MA 02780

Re: Cryptography Technology

Dear Ted:

This brief disclosure is an example of a clear-signed document with a simulated digital signature (25x13 bits = 325 bits) that resides as a graphic within a signature block that is excluded from the signature calculation of the document. In an actual implementation, the user will place the signature block in the document and will then "print" the document to the digital signature printer driver. The driver will then create a graphic file such as a TIFF (multi-page if necessary), remove the graphics in the region where the signature block will go (setting that graphic data to a default blank value), compute signature for the entire document except the signature region, and place the signature data in the region as graphic-mode text.

The document's signature can be verified simply by opening it with a customized TIFF reader, which will detect the presence of the signature region and will validate the signature within it against the data of the document except the graphics within the signature block. An option can be provided to put the signature data on an entirely separate page of the document (e.g., after the last page), preferably with a facsimile copy of the signer's ACI. (In such embodiments, the ACI should have a blank space for the signature data of document signed with the ACI's signing key.)

An exemplary process is illustrated in the attached FIG. (AE-4) , in which a signer creates a document (e.g., this letter) using whatever software he (or she) wishes to use (e.g., Microsoft Word 97). He then prints the document to the SelfCertify.com virtual TIFF printer (call it a "software signature machine"?). The printer driver software creates a TIFF file of the document and displays it in a viewer window. The user interface of the viewer window requests the following input from the signer:

1. Selection of a graphical region within the displayed document for application of the signer's digital signature "stamp." The user can specify the region by moving a dashed box around the screen. The dashed box can have left and right arrows within it for navigating to different pages of a multi-page document, and an "Sign Here" or "OK" button for applying the digital signature "stamp" at the

current location of the box. An unsigned signature page of this letter is attached showing what such a box might look like as it is moved around during the selection process.

2. Authentication from the signer to apply his digital signature to the document within the digital signature "stamp" at the selected location. As discussed in other disclosures, the authentication can be a securely delayed passphrase input.

When the signer has selected the location of the digital signature stamp and provided authentication for its creation, the printer driver software removes all graphic information from the selected location. It then computes the digital signature based on (1) the remaining data of the TIFF file (exclusive of the location of the stamp) and (2) the signer's private key.

Advantageously, the signed document is in a conventional format (e.g., multi-page TIFF) that can be read by any conventional viewer is signature authentication is not needed. When signature authentication is needed, a document can be viewed using the SelfCertify.com TIFF viewer (which may be distributed freely to encourage use of SelfCertify.com's digital signature services).

To verify a digital signature, the viewer searches the graphical rendering of the signed document for the distinctive graphical outline of the signature block. Distinctive features of the graphical outline can include (1) a distinctive color such as maroon, (2) a distinctive line shape such as double parallel lines, and (3) a distinctive line weight such as 3.2 points (not an integer or x.5 fraction). All of these distinctive properties are present in the simulated signature block for this document. In addition, the signature block can have a predetermined size to further identify it.

Once the viewer software has identified the exact location of the signature block, it removes all of the graphic data of the signature block (up to and including the border) and sets it to the default blank value used during signature calculation. It performs an optical character recognition of the signature data within the block to obtain the value of the digital signature. According to advantageous aspects of the invention disclosed elsewhere, the digital signature can be applied to a secure delay to obtain another, larger digital signature value. The digital signature value (as shown or as expanded through a secure delay) is then validated against the modified TIFF representation of the document.

Very truly yours,

-SelfCertify.com Digital Signature-
1410-2708-9007-5781-6673
7701-3376-8745-3789-9100
7440-2788-7101-4089-2409
5420-2178-9001-7680-1477
7731-3376-2645-3789-9105

Edwin A. Suominen

NOTE: Signature data can be in the form of a bar code (1D or 2D). If license to PDF format could be obtained, could include signature characters as data (not image pixels) and search would be very simple search for key tag characters in file.

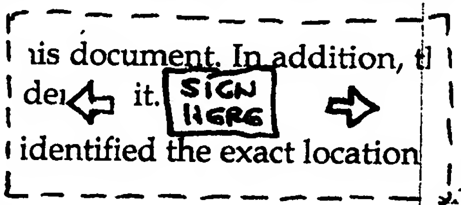
current location of the box. An unsigned signature page of this letter is attached showing what such a box might look like as it is moved around during the selection process.

2. Authentication from the signer to apply his digital signature to the document within the digital signature "stamp" at the selected location. As discussed in other disclosures, the authentication can be a securely delayed passphrase input.

When the signer has selected the location of the digital signature stamp and provided authentication for its creation, the printer driver software removes all graphic information from the selected location. It then computes the digital signature based on (1) the remaining data of the TIFF file (exclusive of the location of the stamp) and (2) the signer's private key.

Advantageously, the signed document is in a conventional format (e.g., multi-page TIFF) that can be read by any conventional viewer is signature authentication is not needed. When signature authentication is needed, a document can be viewed using the SelfCertify.com TIFF viewer (which may be distributed freely to encourage use of SelfCertify.com's digital signature services).

To verify a digital signature, the viewer searches the graphical rendering of the signed document for the distinctive graphical outline of the signature block. Distinctive features of the graphical outline can include (1) a distinctive color such as maroon, (2) a distinctive line shape such as double parallel lines, and (3) a distinctive line weight such as 3.2 points (not an integer or x.5 fr :operties are present in the simulated signature block for iis document. In addition, tl signature block can have a predetermined size to furthe de it.



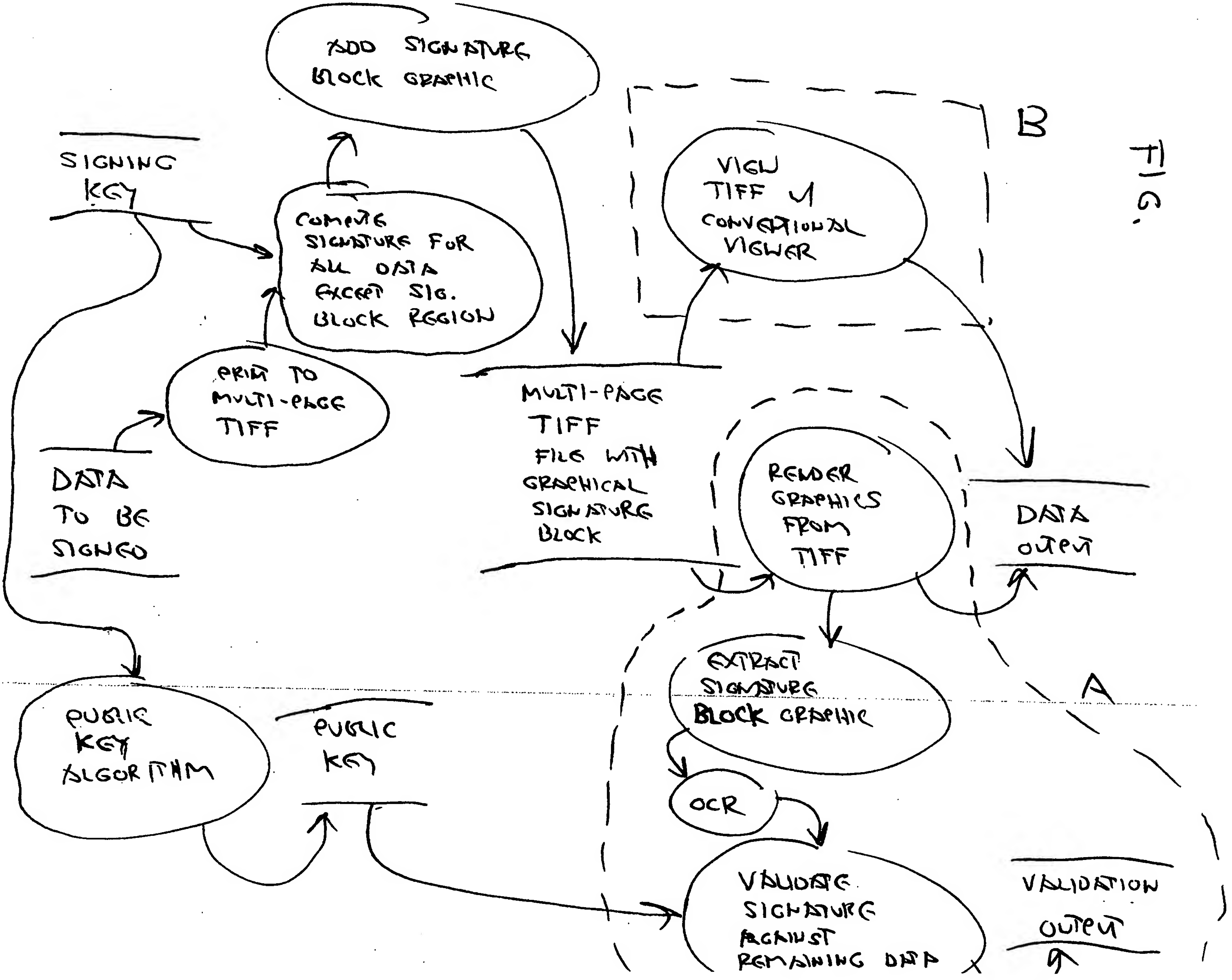
Once the viewer software ha identified the exact location the signature block, it removes all of the graphic data of ıncluding the border) and sets it to the default blank value used during signature calculation. It performs an optical character recognition of the signature data within the block to obtain the value of the digital signature. According to advantageous aspects of the invention disclosed elsewhere, the digital signature can be applied to a secure delay to obtain another, larger digital signature value. The digital signature value (as shown or as expanded through a secure delay) is then validated against the modified TIFF representation of the document.

Very truly yours,

Edwin A. Suominen

AE-3

FIG.


ADD SIGNATURE BLOCK GRAPHIC
SIGNING KEY
COMPUTE SIGNATURE FOR ALL DATA EXCEPT SIG. BLOCK REGION
PRINT TO MULTI-PAGE TIFF
DATA TO BE SIGNED
MULTI-PAGE TIFF FILE WITH GRAPHICAL SIGNATURE BLOCK
B
VIEW TIFF W/ CONVENTIONAL VIEWER
RENDER GRAPHICS FROM TIFF
DATA OUTPUT
EXTRACT SIGNATURE BLOCK GRAPHIC
A
OCR
PUBLIC KEY ALGORITHM
PUBLIC KEY
VALIDATE SIGNATURE AGAINST REMAINING DATA
VALIDATION OUTPUT

AE-4

ALTERNATIVE: Conventional TIFF + detached PGP signature (e.g. in zip file

[redacted]

PEERNET.DRV TIFF Driver

Selecting a Driver | Supported Platforms | Common Features | Support | Online Solutions

4.0 online MANUAL

14 day TRIAL 4.0

3.0 online MANUAL

14 day TRIAL 3.0

PEERNET

Enterprise Labeling Suite

Raster Print Driver Products

Free Trials

The Store

Contact Us

Home

PEERNET.DRV TIFF Driver converts any document capable of being printed by a Windows application into high quality serialized or multi-page TIFF images. It is ideal for imaging or archiving applications. It's also a handy file-generation tool for cross-platform article distribution.

TIFF conversion is as fast as printing. Document scanning is obsolete. Paper waste is a thing of the past.

Can use existing technology to print signed documents according to various aspects of the present inventions.

Features of PEERNET.DRV TIFF Driver

Append Mode (Version 4 only)
Build a multi-page file gradually by appending pages to an existing file whenever the need arises.

Color Correction
Lets you adjust the image's appearance to compensate for monitor non-linearity on Windows NT 4.0 and Windows 2000.

Color Modes
Color or "Black and White only".

Color Reduction
Turn automatic color reduction on or off, to suit your needs. Automatic color reduction reduces your image to its fewest number of colors without affecting picture quality. In addition, automatic color reduction selects the optimal output format for serialized files or the optimal output page format for multi-page files.

Compression
Turn compression on or off as needed.

Output Formats
TIFF True Color Uncompressed or Compressed Packbits
TIFF Monochrome Uncompressed or Compressed CCITT Group 4
TIFF 256 Color Uncompressed or Compressed Packbits

Paper Size
Converts custom documents up to 18.03 x 18.03 inches, or 458 x 458 millimeters. Supports most paper sizes.

Resolutions
100, 200, and 300 DPI resolutions.

AE-5

ZDNet DOWNLOADS

- Top 100 Products
- Download The Future
- Free Downloads

ZDNet > Downloads > Utilities > Printer Utilties > **EZ-Printer for Windows NT**

Another example of existing TIFF generati technology that may be adapted for use according to various aspects of the present invention

Search For: [ ] PC Downloads ▾ GO

- Search Tips
- Power Search

Downloads Home

Pick a Category:

Games
Internet
Utilities
Home & Education
Small Business
Graphics & Multimedia
Development Tools
Windows 95/98/NT
Commercial Demos

See Also:

Editors' Picks
ZDNet Exclusives
PCMagazine Utilities

More Downloads:

Macintosh
Windows CE
PalmPilot
LaunchPad

Services:

Help
Upload a File
Free Newsletter
SmartPlanet

EZ-Printer for Windows NT

10-26-99

Print to image file from any application

ZDNet Rating ★★★★ **User Rating** NR

Be One of the First to Rate This Title!

Company: Suntop
Version:
Size: 774.95 KB
Downloads: 485

▶ **Download Now**

Get GOZILLA And Download Better

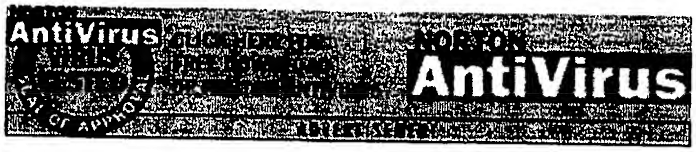
▶ **Add to Basket**
▶ **Look in Basket**

- View file contents

Requirements: Windows NT 4.0
Purchase Information: Demo: $49-$59 for retail version.

EZ-Printer for Windows NT makes it possible to print from any application to an image file. It installs itself as a native device in the printer control panel. Users can simply choose Print and select the EZ-Printer printer from the list of available printers. Output is in black and white (the author has a color version, too) and is automatically saved in the file format of choice (DIB BMP, GIF, TIFF, or PNG). The images are automatically sequentially numbered. The included viewer will let users see the results immediately. This demo version of the driver includes a banner in the middle of the image. Virtually no documentation is included, so users will need to visit the authors' Website to get information or support. Output also seems to be limited in resolution, with no apparent way to control the dpi. Still, EZ-Printer could be particularly useful for Websites, writing documentation, presentations, and many other applications.
Reviewed on Oct 23 1999.

Download Now   Add to Basket



See Related Links

» **Hot Clicks**

- Yahoo! Messenger
- Today's Free File
- Instant Software
- Weekly Top 20

SPECIAL OFFERS from our Partners

Faster Downloads!
It's Free! The Go! Zilla download manager

One-Click Answers
Get GuruNet for simple, fast, relevant information.

TaxACT 2000 Preview
Get a jump start on your 2000 tax return!

Fix your PC for FREE!
Optimize Windows with System Mechanic-try it free!

Protect Your PC
Download Sygate Personal Firewall. It's FREE!

SP i-drive Sideload

E-mail this

Print this

AE-6

Appendix AF

PRINT TO SIGNED TIFF FILE

According to various aspects of the present inventions, a document can be signed by printing the document to a TIFF file using a virtual printer driver, e.g. provided by a service such as "SelfCertify.com." (Note that SelfCertify.com is not an operating business entity, though the applicant has registered the domain name, and has not offered any product or service for sale as of the filing date of the present provisional application.) The TIFF file is created as it normally would be except that includes a signature block within a suitable field.

In one embodiment, the signature block is included within the TIFF "ImageDescription" field, the ASCII contents of which are excluded from the signature calculation of the file. (See AF-3.) In an actual implementation, the user "prints" the document to the digital signature printer driver. The driver will creates the TIFF (multi-page if necessary), with blank characters in the fixed-length "ImageDescription" field where the signature data is intended to reside (setting that data to a default blank ASCII value), compute signature for the entire document with the default blank value in the "ImageDescription" field, and place the signature data in the "ImageDescription" field as ASCII.

The document's signature can be verified simply by opening it with a customized TIFF reader, which extracts the signature data from the "ImageDescription" field and validate it against the data of the document with default blank values substituted in the "ImageDescription" field.

An exemplary process is illustrated in the attached FIG. (AF-4), in which a signer creates a document using whatever software he (or she) wishes to use (e.g., Microsoft Word 97). He or she then prints the document to the SelfCertify.com virtual TIFF printer. The printer driver software creates a TIFF file of the document and displays it in a viewer window. The user interface of the viewer window requests authentication from the signer to apply his digital signature to the document. The authentication can be a securely delayed passphrase input according to various aspects of the present inventions.

Advantageously, the signed document is in a conventional format (e.g., multi-page TIFF) that can be read by any conventional viewer is signature authentication is not needed. When signature authentication is needed, a document can be viewed using the SelfCertify.com TIFF viewer (which may be distributed freely to encourage use of SelfCertify.com's digital signature services).

The signer's public key can be included in the signature data along with the digital signature and the name of the signer. A facsimile copy of the key ACI (see, e.g., Appendix A) can be appended to the end of the TIFF file after the document pages.

Having both of these items present in the TIFF file permits a verifier to quickly authenticate the signed document. If he or she is satisfied with the integrity of the TIFF reader/verification software he or she has obtained (e.g., from a secure, trusted web page of SelfCertify.com), and if he or she is satisfied that the facsimile copy of the ACI with its security background digits is not a forgery, he will have everything he needs to validate the signature of a person who has not made any prior digitial assigning arrangements with him.

Signing and verification software according to various aspects of the present inventions can be integrated with the GPG ("GNU Privacy Guard") software, which can be freely distributed and modified under the GNU Public License. The signing and verification software can call the GPG software with parameter passing. The inventor of signing and verification software can thus include, generally, a slightly modified TIFF printer driver and TIFF reader that calls the GPG software for all digital signature functionality. All three pieces of software can be released in a single compact package.

The TIFF reader software can provide the option to output the original TIFF file (without signature data in the "ImageDescription" field) along with a PGP-compatible detached signature to the file and an ASCII file with the signer's public key. This permits less trusting users to verify integrity of the signature and signing key (comparing PGP's SHA-1 "fingerprint" to what's shown on the ACI) with their own copy of PGP or GPG.

# # #

Sample "ImageDescription" Tiff Field

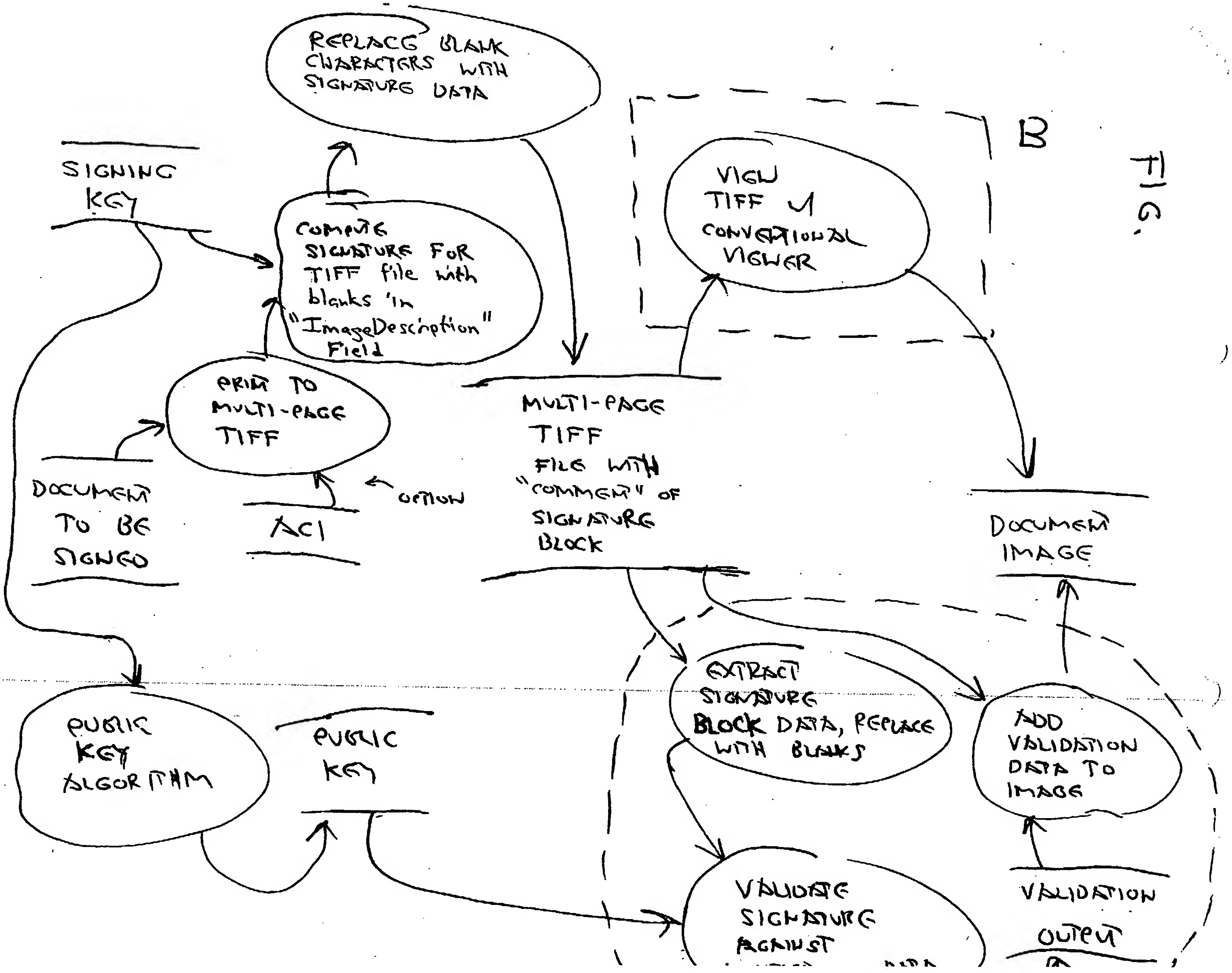
```
Tag = 270
Type = ASCII
Count = Fixed number of characters in signature block, plus 1 for NUL at
end.

Value = 1728
```

```
"-- SELFCERTIFY.COM SIGNATURE DATA -- ↵
Signed by: Edwin A. Suominen ↵
mQGiBDk8ZlMRBADZZ0behMne0qwL7mu7fa/KfbPx2wLtMSihh3IitOo6o6e/twYQ ↵
3Z27YlIlu9uvhIkdsBrQ7b+N0paKyJAu691eE5gzP8VEdzfLJtCQDXvdO9+H57Er ↵
PGicVujuGGIPxvzA7QuyxNxDzndKtFIGO60zn452pWrg/77iA+Ne0CYCuQCg/9I3 ↵
6bNa0vfxxUV3CS+/PDo9VpED/jwquHOyJQYOi0jZNdaT9ZN8mzRQlPgfyGHuNBpp ↵
yoPRhck0nMe1BxNG83M2v243M0DUmabCPuGqqOtYKe5YLqAw/iM5IWwp3EctEHGU ↵
48r8gC7rYKRHOLosyBfx6/uQkpGiNeM4TAI9QpqVzsbHvF1H5LSauLbHOSAwTUoM ↵
e/+uBACTQjV15XA+MPwaIx1vZ31D2iEX/XFLOedxA5XzcN9uVlet+Chxd7xJn66x ↵
w0xzdVLvb/kCdcNY8idJkJVqAUx249S+PymCQFR+sX0pxXCVky4JgtLDToX0wW1G ↵
6C0kPzUR1AH9jdAUaPc+7SC1TdixmOPsLR1l+5PzUUVhNj6b8bQgRUFTIEp1bmUg ↵
MjAwMCA8ZWRAZWVwYXRlbnRzLmNvbT6JAFQEEBECABQFAjk8ZlMFCQAiRwAECwMC ↵
AQIZAQAKCRCRZ8BksUXUY2Y3AKCG99iXRgxGmOssyOC0Lwm/U0yECACfW6R9rI2f ↵
G+UeNOWE/b2TJDt49La5Ag0EOTxmUxAIAPZCV7cIfwgXcqK61qlC8wXo+VMROU+2 ↵
8W65Szgg2gGnVqMU6Y9AVfPQB8bLQ6mUrfdMZIZJ+AyDvWXpF9Sh01D49Vlf3HZS ↵
Tz09jdvOmeFXklnN/biudE/F/Ha8g8VHMGHOfMlm/xX5u/2RXscBqtNbno2gpXI6 ↵
1Brwv0YAWCvl9Ij9WE5J280gtJ3kkQc2azNsOA1FHQ98iLMcfFstjvbzySPAQ/Cl ↵
WxiNjrtVjLhdONM0/XwXV0OjHRhs3jMhLLUq/zzhsSlAGBGNfIShCnLWhsQDGcgH ↵
KXrKlQzZlp+r0ApQmwJG0wg9ZqRdQZ+cfL2JSyIZJrqrol7DVekyCzsAAgIH/0V8 ↵
DY5pj51RDGsakRhMebL9Ob7v9GsbZN6PfTgO2upuCi6WUyazabw4J4ZFc7vtpo8x ↵
FQOkCofOLmisNim7r0PyWrW0SgHLbcXwMMUUb1h/QbggH0WtkkJTzxgNGL+MLJZa ↵
ND4R0gle03PQep4SZgA6/x9OUGWStmzWEt3jk/VdnImS5gDJmNHmCX7+ZaCxROiL ↵
zO3oDmzIRpVYk3+tnekDVhhrDwX5lQ1zUoCg43hAmA1Q1/KNFBw/qiol0EvLyJby ↵
hUzhGqdzd/MJkNHXviOoJyuOnQH+O81EME5S2Ej19epf4Rfuf9rh8uR7tl3YEraD ↵
wqwO4VIcd5n+6F3199GJAEwEGBECAAwFAjk8ZlMFCQAiRwAACgkQkWfAZLFF1GMG ↵
KwCZARTQgJDOM40GBp0OJwP1escVP/gAoPIJb/gmbNpbeQmG9UobWiT8PKl1 ↵
=zFrB ↵
↵
Signature: ↵
iQA/AwUAOhs1jKmKuMvNCWDGEQJ4TACeJpwTCOzNvxKhZVYag171BEuhKEMAnjtT ↵
SivKAZgC21P/pMrro2HgTfJo ↵
=Adug<NUL>"
```

AF-3

FIG.


B
SIGNING KEY
REPLACE BLANK CHARACTERS WITH SIGNATURE DATA
COMPUTE SIGNATURE FOR TIFF file with blanks 'in "ImageDescription" Field
PRINT TO MULTI-PAGE TIFF
DOCUMENT TO BE SIGNED
ACI
OPTION
MULTI-PAGE TIFF FILE WITH "COMMENT" OF SIGNATURE BLOCK
VIEW TIFF W/ CONVENTIONAL VIEWER
DOCUMENT IMAGE
EXTRACT SIGNATURE BLOCK DATA, REPLACE WITH BLANKS
ADD VALIDATION DATA TO IMAGE
VALIDATE SIGNATURE AGAINST
VALIDATION OUTPUT
PUBLIC KEY ALGORITHM
PUBLIC KEY

AF-4

IMPROVEMENT TO PGP
By Edwin A. Suominen

Allowing keys of keyring to exist at different locations

I would like to follow the advice of some encryption experts to keep my private key (or at least a signing key) on removable media (e.g., a floppy disk) for added security. However, the only way to do this now is to have my entire PGP keyring on that floppy disk. If the key I'm truly worried about is to be secure, I won't want to keep that floppy disk where it is easily accessible. Instead, I will want to keep my public keys and my encryption key on the hard disk and just my signing private keying on a floppy disk.

According to this proposed improvement, PGPkeys allows users to specify the file location of private keys on their keyring. Everything works normally except when an "indirect" private key is to be used, e.g., to apply an especially important PGP signature to a contract. When that happens, the PGP software looks for the file containing the indirect key. If the file is located on removable media that is not in the drive, the software prompts the user to insert the removable media containing the indirect key. If the file is located on a PGP disk or some other type of encrypted volume, the software could simply indicate that the volume is not present and invite the user to mount it.

After use of the PGP key is complete, the software can advise the user that it is no longer needed and that he or she can unmount the volume or put the removable media back in its secure location.

AG-1

To:
From: Ed Suominen <ed@eepatents.com>
Subject:
Cc:
Bcc: ed@eepatents.com
Attached:

LAW OFFICES OF LOUIS J. HOFFMAN, P.C.
14614 North Kierland Boulevard, Suite 300 * Scottsdale, Arizona 85254
Telephone: (480) 948-3295 * Facsimile: (480) 948-3387

Edwin A. Suominen * Admitted to practice in patent matters before the U.S. Patent Office only
Web Site: http://eepatents.com * PGP Public Key: http://eepatents.com/key

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1
```

This is an example of a message that has been signed with **preserved formatting** and an *unobtrusive* digital signature around clear-signed text, according to various aspects of the present inventions.

```
-----BEGIN PGP SIGNATURE-----
Version: PGP Personal Privacy 6.5.8

iQA/AwUBOqO3eqmKuMvNCWDGEQLwpwCePj1y0iuPEKIeRsSyqTCA7S++MpIAnRPv
qtttmsePjh/WqGafymg/hVMs
=q4Oy
-----END PGP SIGNATURE-----
```

AH-1

CRYPTOGRAPHIC DOCUMENT DESTRUCTION

BACKGROUND AND SUMMARY

In private confidential conversation, two people can have a conversation without leaving any record of their conversation. With written or electronic communications, however, there is some record of what was said. That record can be difficult to eliminate.

Paper communications such as letters can be shredded if both sender and recipient agree that they will destroy their copies. Electronic communications (e-mail) are more difficult to eliminate because backup copies can be made and automatically archived onto other locations. It is sometimes surprising that backup copies are available during discovery of communications that would be embarrassing. Accordingly, there is a need for a system of destroying electronic communications or records when the sender and recipient of the communications agree to do so.

A system according to various aspects of the invention includes: an encryption subsystem; and a decryption subsystem, the decryption subsystem using a temporary key that can be disposed of to make encrypted communications unreadable.

DETAILED DESCRIPTION

An encryption key allows an authorized person to decrypt encrypted communications. For example, an encryption key can be a passphrase, or use a passphrase, known only to a person authorized to decrypt communications. According to various aspects of the present inventions, the decryption key can be destroyed. A passphrase for such a decryption key is preferably forgettable, for example a random alphanumeric string of sufficient length to be secure. Advantageously, the alphanumeric string can be used as a passphrase to open communications or records when is desired to do so and then destroyed and forgotten about when it is no longer desired for such communications or records to ever be decrypted again.

Operation of one embodiment includes (1) writing an electronic mail message to a person; (2) encrypting the communications using an agreed-upon passphrase, preferably an alphanumeric random digit string, for example 12 digits in length; (3) sending the encrypted message to the recipient; (4) having the recipient type in the passphrase to open the encrypted communications; and then after a predetermined or agreed-upon period of time, (5) having both parties destroy the passphrase (throwing away a Post-It note upon which the passphrase is written) so that neither the sender nor the recipient can ever decrypt the communication again. Preferably, the passphrase is used only for a short length of time or limited number of times so that it is impossible for either party to remember it. The more random and arbitrary cryptic the passphrase is, the more difficult it will be to ever remember.

Systems according to various aspects of the invention can be useful in the legal profession where sometimes legal professionals are called upon to testify about matters that were assumed to be privileged but the court determines that they are not for whatever reason, as happens in patent practice sometimes. If an attorney or agent has communicated with his client using this system, and the client agrees to destroy the passphrase after the matter is complete, and the device communicated by the attorney or agent is no longer relevant or needed and has been acted upon completely, then it is impossible for any court or any party to discover what to parties discussed.

Even if a backup copy of an electronic mail message is found, a court can authorize a cryptananalysis of the message, but if it is encrypted using PGP strong encryption, it would be very difficult, effectively impossible, for the opposing party to figure out what the message said.

Embodiments can be employed in other types of communications that are encoded in digital form so that they can be encrypted. Even handwritten notes can be scanned into digital form and encrypted.

Voice messages can be digitized and compressed, entire paper files can be archived by scanning and digitizing, and then encrypting into a single encrypted archive file with a temporary key that can be disposed of after a predetermined time, which can be set by policy, for example one year.

According to another aspect of the invention, the keys need not be remembered or typed in by a human operator at all. According to this aspect, the key is an actual hardware device that transmits decrypting authorization indicia for a predetermined or agreed-upon period of time and then is incapable of doing so after that. An example of such a key is placed between a conventional PC keyboard and a PC. The device includes circuitry for reading a decryption key code or indicia from another device such as a card having barcodes printed on it or a disposable integrated circuit, which can be made in the form of a key.

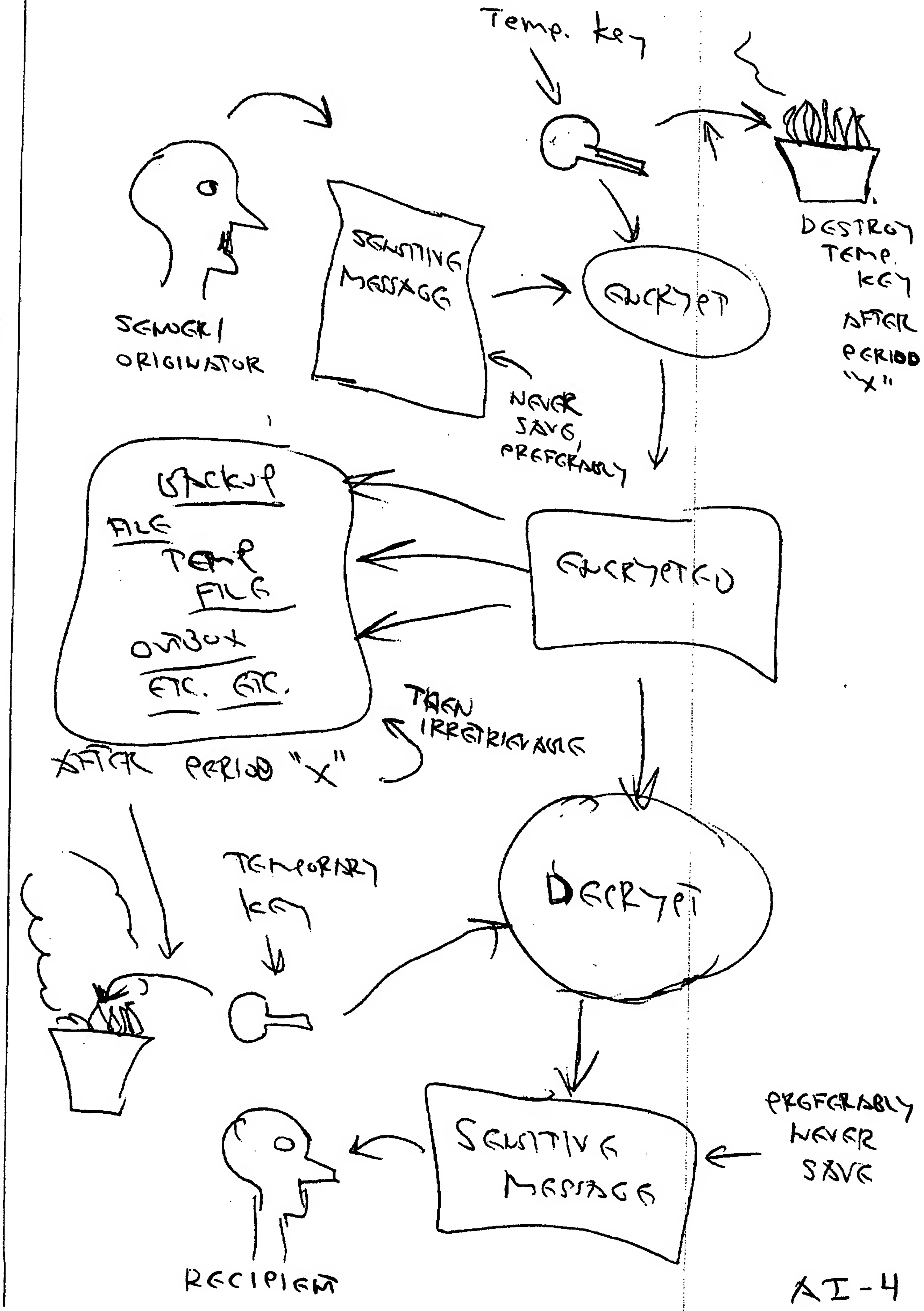
The device can be sold with a number of keys that can be used and disposed of by the user. For example, if the device is sold with twelve keys with refills of 12 additional keys available by ordering, the user can encrypted archive records every month with a different key and cost a new (different) key away every month. The user may wish to keep the records on file for a period of several months in which case the user will begin using a key one month and then put the key into storage for a couple of months and then toss the key away, destroying it irretrievably after that period of time.

An embodiment using printed cards is less expensive and the keys can be disposed of more cheaply but it is more prone to unauthorized or inadvertent duplication, in which case the whole purpose of the system might be defeated. The user of such a system needs to take precautions that the keys are never duplicated.

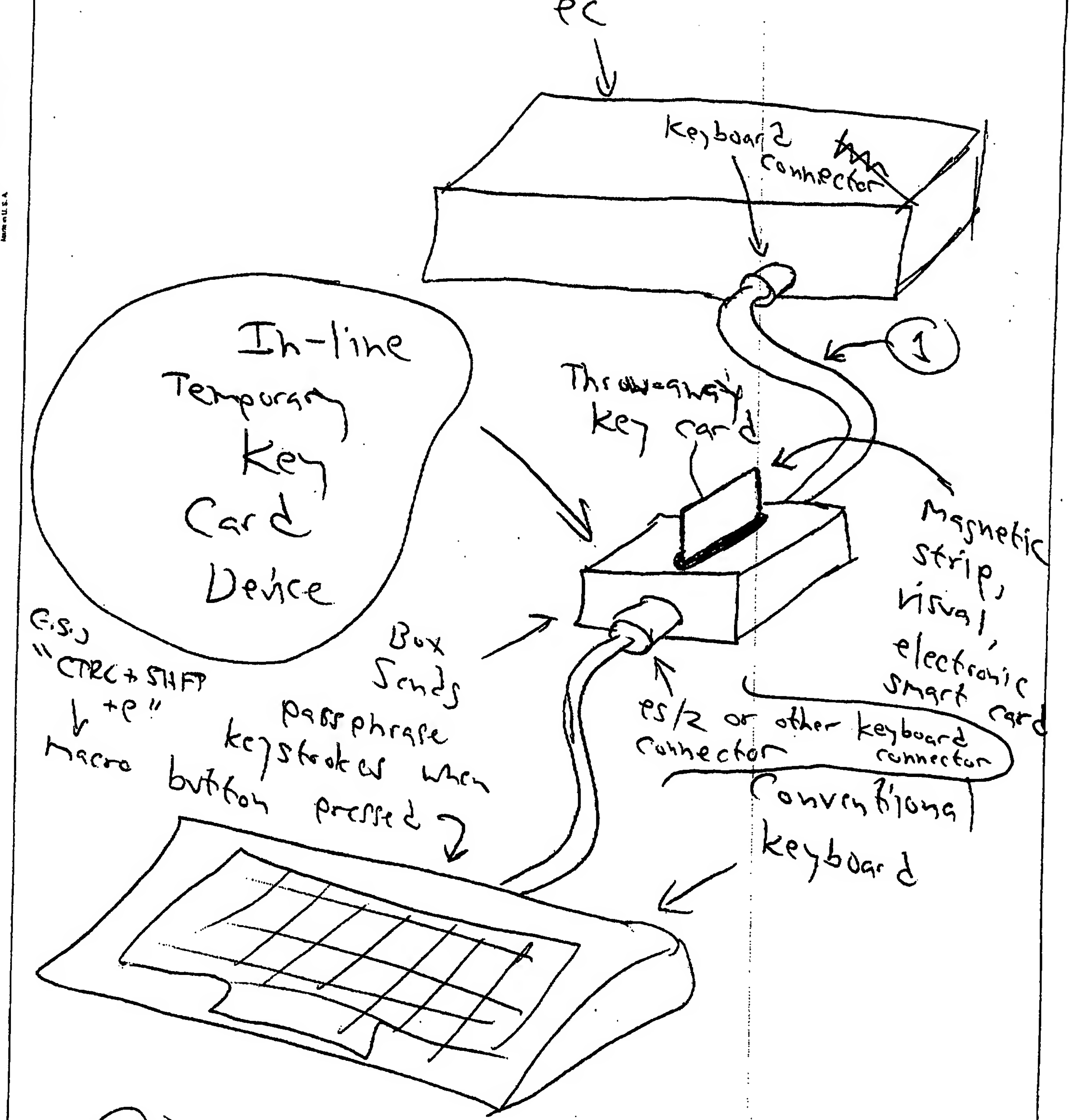
A database of which files correspond to which temporary keys can be created according to aspects of the invention so that an administrator can look over the list of keys about to expire and ask the persons involved with the effective files whether or not they need information from the files before they are destroyed. Paper documents can be shredded at the same time the keys are destroyed. If the key for a paper file that has a corresponding electronic file is a card, the card can be kept with the paper file and both can be destroyed simultaneously.

The key can be distributed from a sender of information to a recipient who is only authorized to access the information for a temporary period of time, for example one or two days. The sender of the information, or provider, can demand the key back after that period of time. In such a system, the key needs to be difficult to duplicate, for example an integrated circuit in the shape of a key. A forgettable password would not work for such a system because the user could write it down without telling the sender, but the forgettable password system works well when both parties, or all parties involved, are in cooperation and consent to destroy the information and the forgettable password.

# # #


Temp. key
DESTROY TEMP. KEY AFTER PERIOD "X"
SENDER / ORIGINATOR
SENSITIVE MESSAGE
ENCRYPT
NEVER SAVE, PREFERABLY
BACKUP
FILE
TEMP FILE
OUTBOX
ETC. ETC.
ENCRYPTED
THEN IRRETRIEVABLE
AFTER PERIOD "X"
TEMPORARY KEY
DECRYPT
PREFERABLY NEVER SAVE
SENSITIVE MESSAGE
RECIPIENT

AI-4


PC
Keyboard connector
In-line Temporary Key Card Device
Throwaway key card
1
Magnetic strip, visual, electronic smart card
Box Sends passphrase keystrokes when macro button pressed
e.g., "CTRL + SHFT +P"
PS/2 or other keyboard connector
Conventional keyboard

1 Can be tamper alarmed, won't send pass-phrase ("key") keystrokes if ~~tap~~ is detected

AJ-1

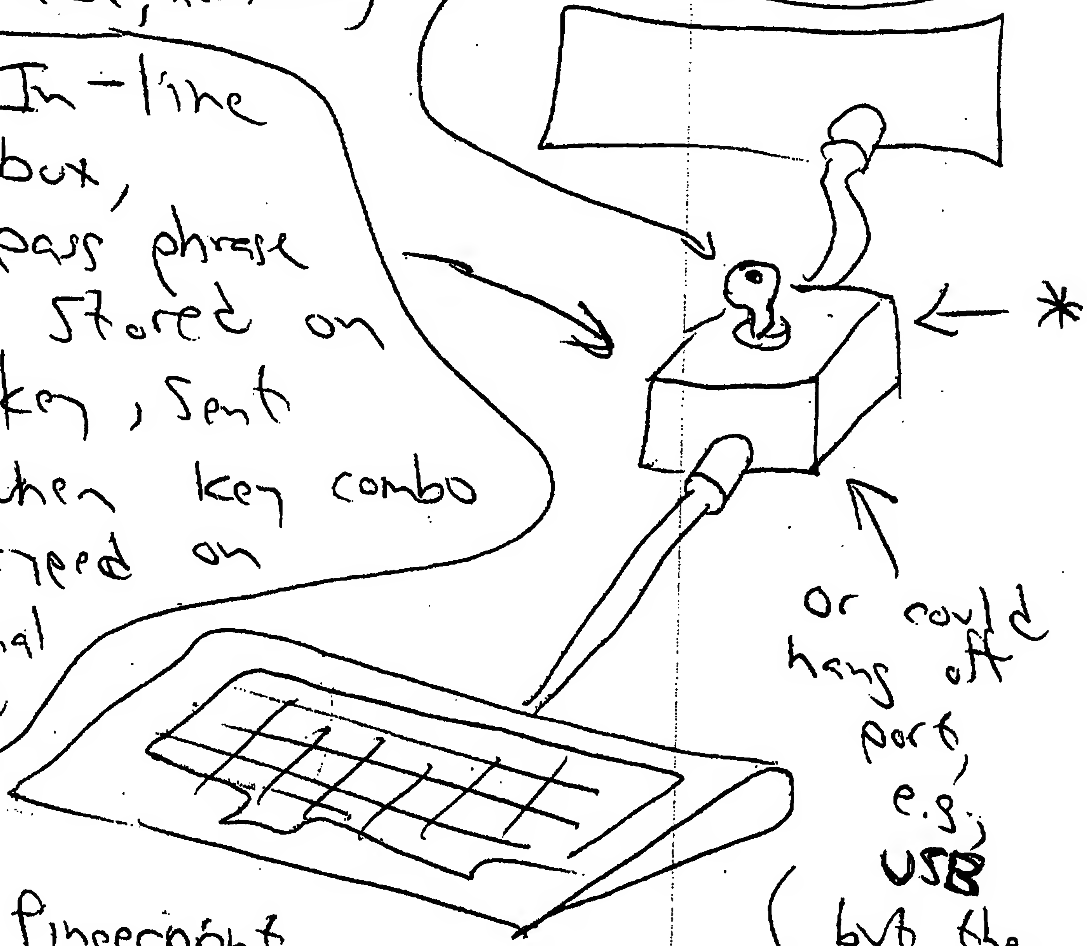
preferabl, w/ secure delay processor in box here and AJ-1

(can also be used with permanent key systems)

physical "private key" for conventional crypto. systems (e.g., PGP, hushmail)

In-line box, pass phrase stored on key, sent when key combo typed on conventional keyboard

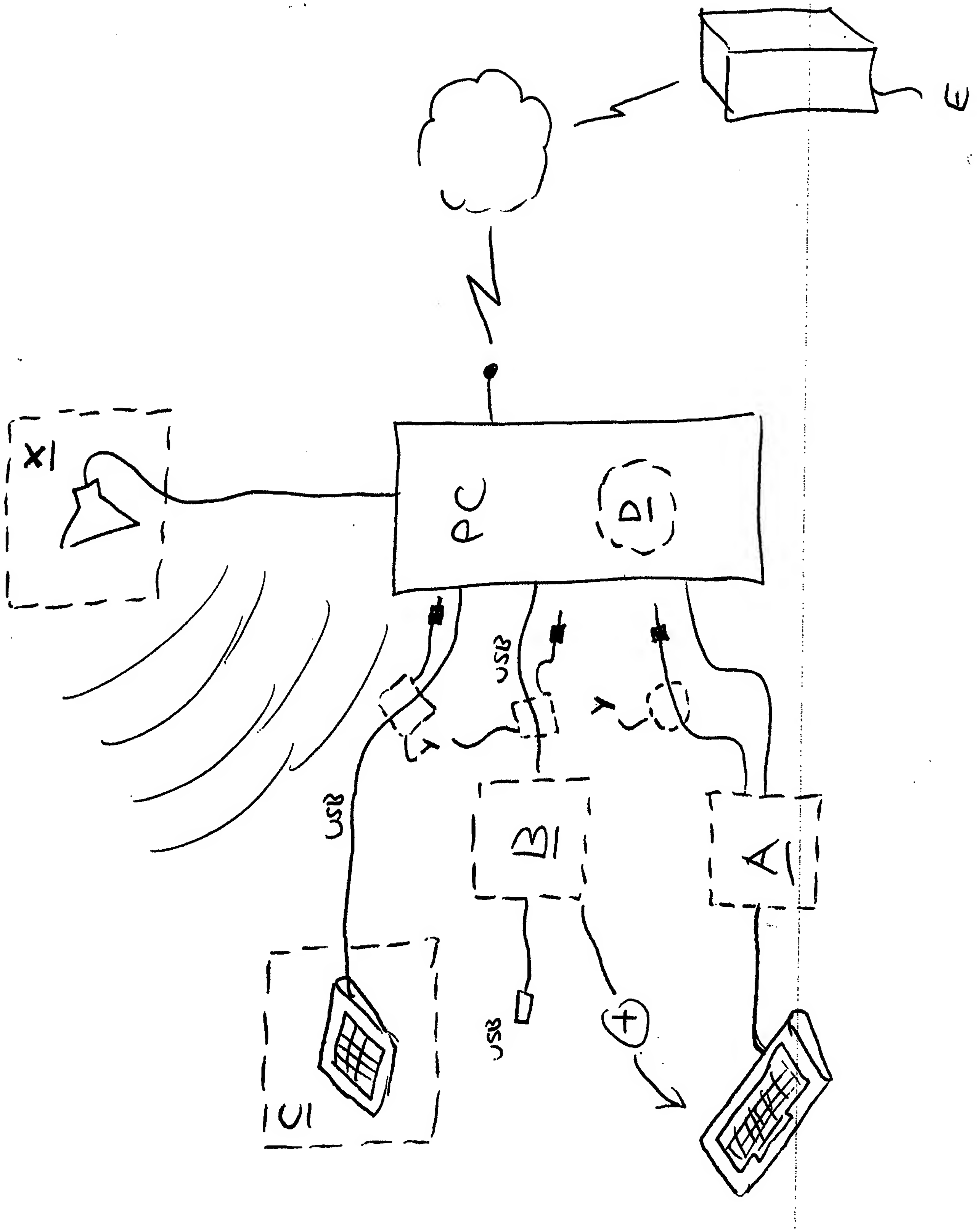
or magnetic card



or could hang off port, e.g., USB (but then raises question of sniffing ports)

* or fingerprint (or other biometric) sensor, w/ patterns as "passphrase" or "key"

AJ-2



AJ-3

CLOCK JITTER RANDOM BIT GENERATOR

This invention uses existing hardware in a standard desktop PC with some very compact software to efficiently generate truly random numbers. Tell that introductory sentence to any cryptographer and they will either (1) beg you tell them how, or (2) dismiss you as an ignorant crackpot and lead you to various references talking about detecting hard disk read latencies, digitizing thermal noise, counting bubbles in fish tanks or lava lamps for generating true random numbers. (Pentium III chips have a hardware random number generator embedded in them, but I'm not sure that the AMD chips do. In any event, this invention provides random numbers from hardware that exists in all PCs today, and the need is still out there for a simple software solution.) I think if they were to read the following paragraphs they would scratch their head and say "Now why didn't I think of that?"

A method of generating a random binary bit according to various aspects of this invention includes: (a) clocking a CPU using a phase-locked loop; (b) beginning a count of CPU clock cycles upon a first transition of an independent clock signal (e.g., the PC's real-time clock, which generates interrupts on IRQ08 under control of a 32.768 kHz crystal); (c) recording the number of CPU clock cycles counted upon a second transition of the independent clock signal; and (d) extracting the least significant bit of the count to serve as a random binary bit.

Modern microprocessors use high-frequency clocks that are generated using a phase-locked loop (PLL). The PLL multiplies a lower-frequency crystal oscillator clock signal to generate microprocessor clock, as well as other clock signals used in the system. The Intel "CK98 Clock Synthesizer/Driver" is an example of a chip containing a PLL for this purpose.

PLL's lock the output frequency of a relatively unstable voltage-controlled oscillator to the more stable reference frequency of a crystal oscillator. The locking is performed using a feedback loop that has particular characteristics. Because phase locking cannot be perfect, there is always some "phase noise" or jitter on the PLL output. This phase noise or jitter causes the frequency of the PLL output signal to randomly vary within a Gaussian distribution. CPU clock generators are not designed to have very low phase noise because the exact clock frequency of the CPU is not particularly important. (That is in contrast to the type of PLL design I'm familiar with, for cellphone receivers, where the phase noise needs to drop off within a few hundred Hz of the carrier.) The PLL of the CK98 chip, for example, specifies that the "ideal closed loop jitter bandwidth" be attenuated 20 DB at 500 kHz, and a graph in the chip's data sheet shows the jitter (phase noise) spectrum

AK-1

extending out to 50 kHz without significant attenuation.

With these levels of jitter, the number of CPU clock cycles in any given time interval can be expected to have significant variance. The high frequency components of the phase noise will integrate over long intervals and cancel each other out to some extent, but the number of CPU clock cycles over one predetermined time interval will be different (in a random fashion) from the number of CPU clock cycles over another predetermined time interval of the same length.

As a time interval gets longer, the number of clock cycles over the interval increases and the possibility of an integer difference between the number of clock cycles in two separate intervals of the same length increases proportionately. However, longer intervals permit high frequency variations in instantaneous frequency to cancel each other out.

The probability of any given frequency measured over a given time interval is defined as a Gaussian function of the frequency offset from the correct PLL frequency. Thus the probability of any given offset frequency measured over a larger time interval can be expected to be smaller than the probability of that frequency over the shorter time interval, by a scaling factor inversely proportional to the square root of the ratio between intervals. So the probability P of an offset frequency fdev measured over an interval T is a function Pf_dev=P(f_offset)/SQRT(T/C). If the interval is quadrupled, the probability of measuring a given offset frequency over that longer interval is 1/2 the probability of measuring that frequency over the shorter interval.

However, the number of clock cycles employed in the frequency measurement increases linearly with the length of the interval. The measured frequency deviation (over the interval) represented by a single LSB of difference in clock cycles is inversely proportional to the number of clock cycles. So, longer time intervals permit more accurate measurement of a given frequency deviation. If an interval is quadrupled, the probability of a given offset frequency difference resulting in an LSB difference is four times the probability at the shorter interval.

In view of the above two paragraphs, it is clear that any amount of random frequency deviation can be measured so long as the interval is made long enough. The probability of measuring a given offset will decrease, but the ability of the measurement to detect that offset will increase faster than the decrease in probability of offset. In addition to this rather crude analysis, an additional factor may be the presence of low-frequency deviations in the Gaussian distribution. It is possible that the integration of the area under the Gaussian curve from zero offset to 1/fdev will increase to the point where a value of fdev=1/T will become likely.

The existence of a PLL to control the CPU clock and a separate crystal oscillator to control a real-time clock in every PC (and probably Macintosh) out there makes this method an extremely useful way to generate truly random bits.

FAQ
Q1: What about other pseudorandom variations caused by other interrupts, which may slow down the count in a predictable fashion?
A1: Uncorrelated variances add, so the presence of any truly random component in a binary number will result in a random LSB has long as the truly random component is at least one LSB in size.

AK-2

Q2: How large would the count of CPU clock cycles be with a Pentium III running at 500 MHz and a 32.768 kHz interrupt rate from the real-time clock?
A2: 15,258.8

Q3: Obviously, you can't measure 0.8 clock cycles. You probably can't measure integer clock cycles either because of the count routine requiring multiple clock cycles. So what sort of resolution could you expect in that count?
A3: Probably a count by 3's, or 5086 possible count values. One for the increment, one for the branch back, one for unknown possibilities.

Q4: So what is the minimum frequency deviation (in Hertz) you could measure over one 30.5 microsecond interrupt interval from the real-time clock?
A4: 98 kHz.

Q5: What if the frequency offset is unlikely to be that large over any given interval, say only a 10% probability?
A5: Accumulate 100 potentially random bits over 100 real-time clock intervals. The chance of every single one of those intervals *not* producing a random bit is 0.90^100 = 0.0027%. Since at least one of the bits will almost certainly be random, simply XOR all the bits together and you'll have a random bit. Note that the specifications of the CK98 clock generator make it seem more likely than 10 percent that you'll see frequency deviations of 98 kHz.

If you actually use 200 intervals because of the need to set up before the next counting interval, the whole process would take six milliseconds per random bit produced.

14.51 Note (*computational efficiency of reduction modulo $b^t - c$*)

(i) Suppose that x has $2t$ base b digits. If $l \leq t/2$, then Algorithm 14.47 executes step 2 at most $s = 3$ times, requiring 2 multiplications by c. In general, if l is approximately $(s-2)t/(s-1)$, then Algorithm 14.47 executes step 2 about s times. Thus, Algorithm 14.47 requires about sl single-precision multiplications.

(ii) If c has few non-zero digits, then multiplication by c will be relatively inexpensive. If c is large but has few non-zero digits, the number of iterations of Algorithm 14.47 will be greater, but each iteration requires a very simple multiplication.

14.52 Note (*modifications*) Algorithm 14.47 can be modified if $m = b^t + c$ for some positive integer $c < b^t$: in step 2.2, replace $r \leftarrow r + r_i$ with $r \leftarrow r + (-1)^i r_i$.

14.53 Remark (*using moduli of a special form*) Selecting RSA moduli of the form $b^t \pm c$ for small values of c limits the choices of primes p and q. Care must also be exercised when selecting moduli of a special form, so that factoring is not made substantially easier; this is because numbers of this form are more susceptible to factoring by the special number field sieve (see §3.2.7). A similar statement can be made regarding the selection of primes of a special form for cryptographic schemes based on the discrete logarithm problem.

14.4 Greatest common divisor algorithms

Many situations in cryptography require the computation of the greatest common divisor (gcd) of two positive integers (see Definition 2.86). Algorithm 2.104 describes the classical Euclidean algorithm for this computation. For multiple-precision integers, Algorithm 2.104 requires a multiple-precision division at step 1.1 which is a relatively expensive operation. This section describes three methods for computing the gcd which are more efficient than the classical approach using multiple-precision numbers. The first is non-Euclidean and is referred to as the *binary gcd algorithm* (§14.4.1). Although it requires more steps than the classical algorithm, the binary gcd algorithm eliminates the computationally expensive division and replaces it with elementary shifts and additions. Lehmer's gcd algorithm (§14.4.2) is a variant of the classical algorithm more suited to multiple-precision computations. A binary version of the extended Euclidean algorithm is given in §14.4.3.

14.4.1 Binary gcd algorithm

14.54 Algorithm Binary gcd algorithm

INPUT: two positive integers x and y with $x \geq y$.
OUTPUT: $\gcd(x, y)$.

```
1. $g \leftarrow 1$.
2. While both $x$ and $y$ are even do the following: $x \leftarrow x/2$, $y \leftarrow y/2$, $g \leftarrow 2g$.
3. While $x \neq 0$ do the following:
   3.1 While $x$ is even do: $x \leftarrow x/2$.
   3.2 While $y$ is even do: $y \leftarrow y/2$.
   3.3 $t \leftarrow |x - y|/2$.
   3.4 If $x \geq y$ then $x \leftarrow t$; otherwise, $y \leftarrow t$.
4. Return($g \cdot y$).
```

H.A.C.

AL-1

14.3.4 Reduction methods for moduli of special form

When the modulus has a special (customized) form, reduction techniques can be employed to allow more efficient computation. Suppose that the modulus m is a t-digit base b positive integer of the form $m = b^t - c$, where c is an l-digit base b positive integer (for some $l < t$). Algorithm 14.47 computes $x \bmod m$ for any positive integer x by using only shifts, additions, and single-precision multiplications of base b numbers.

14.47 Algorithm Reduction modulo $m = b^t - c$

```
INPUT: a base $b$, positive integer $x$, and a modulus $m = b^t - c$, where $c$ is an $l$-digit base $b$ integer for some $l < t$.
OUTPUT: $r = x \bmod m$.
  1. $q_0 \leftarrow \lfloor x/b^t \rfloor$, $r_0 \leftarrow x - q_0 b^t$, $r \leftarrow r_0$, $i \leftarrow 0$.
  2. While $q_i > 0$ do the following:
      2.1 $q_{i+1} \leftarrow \lfloor q_i c/b^t \rfloor$, $r_{i+1} \leftarrow q_i c - q_{i+1} b^t$.
      2.2 $i \leftarrow i + 1$, $r \leftarrow r + r_i$.
  3. While $r \geq m$ do: $r \leftarrow r - m$.
  4. Return($r$).
```

14.48 Example (*reduction modulo $b^t - c$*) Let $b = 4$, $m = 935 = (32213)_4$, and $x = 31085 = (13211231)_4$. Since $m = 4^5 - (1121)_4$, take $c = (1121)_4$. Here $t = 5$ and $l = 4$. Table 14.9 displays the quotients and remainders produced by Algorithm 14.47. At the beginning of step 3, $r = (102031)_4$. Since $r > m$, step 3 computes $r - m = (3212)_4$. □

i	$q_{i-1}c$	q_i	r_i	r
0	–	$(132)_4$	$(11231)_4$	$(11231)_4$
1	$(221232)_4$	$(2)_4$	$(21232)_4$	$(33123)_4$
2	$(2302)_4$	$(0)_4$	$(2302)_4$	$(102031)_4$

Table 14.9: *Reduction modulo $m = b^t - c$ (see Example 14.48).*

14.49 Fact (*termination*) For some integer $s \geq 0$, $q_s = 0$; hence, Algorithm 14.47 terminates.

Justification. $q_i c = q_{i+1} b^t + r_{i+1}, i \geq 0$. Since $c < b^t$, $q_i = (q_{i+1} b^t / c) + (r_{i+1}/c) > q_{i+1}$. Since the q_i's are non-negative integers which strictly decrease as i increases, there is some integer $s \geq 0$ such that $q_s = 0$.

14.50 Fact (*correctness*) Algorithm 14.47 terminates with the correct residue modulo m.

Justification. Suppose that s is the smallest index i for which $q_i = 0$ (i.e., $q_s = 0$). Now, $x = q_0 b^t + r_0$ and $q_i c = q_{i+1} b^t + r_{i+1}$, $0 \leq i \leq s - 1$. Adding these equations gives $x + \left(\sum_{i=0}^{s-1} q_i\right) c = \left(\sum_{i=0}^{s-1} q_i\right) b^t + \sum_{i=0}^{s} r_i$. Since $b^t \equiv c \pmod{m}$, it follows that $x \equiv \sum_{i=0}^{s} r_i \pmod{m}$. Hence, repeated subtraction of m from $r = \sum_{i=0}^{s} r_i$ gives the correct residue.

AL-2]]